112/17526/831
[REDACTED]
6-1-86/46

OK

UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS

**"MÁQUINA E MODELO DE DADOS
DEDICADOS PARA APLICAÇÕES
DE ENGENHARIA"**

Caetano Traina Junior

**Tese de Doutorado apresentada ao Instituto de
Física e Química de São Carlos, USP, para a obtenção
do Título de Doutor em Ciências "Física Aplicada".**

Orientador: Prof. Dr. Jan Frans Willem Slaets

Departamento de Física e Ciência dos Materiais

São Carlos - 1986

BIBLIOTECA DO INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS - USP
FÍSICA

MEMBROS DA COMISSÃO JULGADORA DA TESE DE DOUTORADO DE _____

Caetano Traina Junior

APRESENTADA AO INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS, DA
UNIVERSIDADE DE SÃO PAULO, EM 03 DE dezembro DE 1986.

COMISSÃO JULGADORA:

Dr. Jan F.W. Slaets - Orientador

Dr. Roland Köberle

Dr. Valentin Obac Roda

Dr. Mário Jino

Dr. Fernão Stella de Rodrigues Germano

A Nina, companheira e amiga em todas
as horas, dedico este trabalho.

AGRADECIMENTOS

Ao Prof. Dr. Jan Frans Willen Slaets meu orientador e amigo, que além do empenho para que este trabalho fosse realizado com êxito, sempre encontrou tempo para me ajudar em todas as dificuldades surgidas.

Ao Prof. Alvaro Garcia Neto e ao Prof. Gonzalo Travieso, pela amizade e auxílio no desenvolvimento deste trabalho, e pela fertilidade das idéias originadas de nossas discussões.

Aos amigos Prof. Dr. Mário Jino, Miriam e Fernando Capretz, e Mário Bento de Carvalho, a equipe do Centro Tecnológico para a Informática com a qual trabalhamos no desenvolvimento do SIPS, e que criou o meio para que muitos pontos deste trabalho surgissem e fossem realizados.

A Agma Juci Machado Traina, minha esposa, não só por toda a sua dedicacao a mim, para que eu pudesse executar este trabalho, mas também pela grande colaboração que prestou para a sua implementação.

Aos técnicos do Laboratório de Instrumentação Eletrônica do IFQSC, Lirio, Ailton e Marcos, pelas sugestões e auxílio na elaboração da máquina implementada.

A Maria Benedita, desenhista do Grupo de Instrumentação Eletrônica, pela elaboração dos desenhos apresentados nesta tese e dos diagramas que compõem a documentação do trabalho realizado.

A todas as demais pessoas que, direta ou indiretamente,

contribuíram para a concretização deste trabalho.

Agradeço também ao Instituto de Ciências Matemáticas de São Carlos, que permitiu que eu realizasse este trabalho, e à FINEP e CNPq pelo apoio financeiro.

RESUMO

Esta tese envolve duas áreas de conhecimento, nomeadamente a de modelagem de dados para Sistemas de Gerenciamento de Bases de Dados, e a de desenvolvimento de Máquinas de Base de Dados.

Devido a isso, esta tese apresenta-se também dividida em duas partes. Na primeira parte analisam-se os modelos já existentes e a partir das deficiências que apresentam para aplicações como Bases de Dados para Engenharia, define-se o Modelo de Representação de Objetos. Na segunda parte são analisadas arquiteturas de Máquinas de Base de Dados existentes e faz-se a proposta de uma nova arquitetura dedicada, para suportar uma implementação capaz de aproveitar o paralelismo que o modelo apresentado permite.

Nas duas partes faz-se um levantamento de trabalhos relevantes que existem nas respectivas áreas, e mostra-se como as soluções apresentadas satisfazem as necessidades inerentes de cada parte.

ABSTRACT

This work deals with two research areas: data modeling for Database Management Systems; and development of Database Machines.

Consequently, this work was divided into two parts. In the first part, already existent data models are analysed, and based on the characteristics needed for engineering database applications, the Object Representation Model is defined. In the second part, existing database machines architectures are also analysed, and a new dedicated architecture is proposed, intended to support all the intrinsic parallelism of the presented data model.

A survey of relevant results in both areas is included, and compared with our conclusions.

Contribuições Originais desta Tese

Nesta tese foi elaborado um novo modelo de dados para implementação de bases de dados para engenharia, que tem dentre as suas características, as seguintes contribuições originais:

- O uso de uma base de dados como o meio de armazenagem do esquema de dados da base de dados;
- A introdução do conceito de Classes de Objetos para a identificação de um dado objeto;
- A classificação e o reconhecimento de características de atributos pelo SGBD;
- O uso de um relacionamento para identificar um objeto;
- A introdução do conceito de Para-Relacionamentos que, definidos no esquema de dados, permitem a "herança" de atributos e relacionamentos entre objetos na base de dados.

A viabilidade de aplicação do modelo proposto já foi comprovada com a implementação do Sistema Integrado de Produção de Software - SIPS - em andamento no Centro Tecnológico para a Informática (MCT/CTI).

Foi também proposta uma arquitetura especial para suportar o Modelo de Representação de Objetos, que aproveita o potencial de paralelismo que o modelo apresenta. Dentre suas características são também contribuições originais:

- O uso de um mecanismo lógico de alto nível para se determinar as regiões da base de dados que contém as informações necessárias a cada operação;

- A solicitação de operações ser reconhecida diretamente pelos processadores de manipulação de dados da máquina, através de pedidos de alto nível provenientes da aplicação;
- A existência de um processador que analisa todos os pedidos de operações, para otimizar a distribuição de dados nas memórias dos vários processadores de manipulação.

I N D I C E

I. - INTRODUÇÃO.

I.1. - Considerações Gerais.	1
I.2. - Apresentação.	6

----- Parte I -----

II. - APRESENTAÇÃO DO PROBLEMA.

II.1. - Introdução.	11
II.2. - Os Principais Modelos de Bases de Dados Usados Atualmente.	14
II.3. - Especificação das Necessidades das Bases de Dados para Engenharia.	35
II.4. - Modelos não Convencionais de Interesse em Aplica- ções para Engenharia.	57
II.5. - Conceitos Adicionais Necessários ao Modelo Proposto.	79
II.6. - Conclusões.	95

III. - O MODELO DE REPRESENTAÇÃO DE OBJETOS.

III.1. - Introdução.	97
III.2. - Definição e Modificações em Esquemas de Dados no Modelo de Representação de Objetos.	101
III.3. - A Estrutura de Dados de Acesso à Informação no Modelo de Representação de Objetos.	111

III.4. - Maior Especificidade em Características de Tipos de Atributos.	114
III.5. - Escopo de Abrangência dos Identificadores de Objetos.	121
III.6. - Identificação de Objetos Através de Relacionamentos.	129
III.7. - O Uso do MRD em um Ambiente Multiusuário.	131
III.8. - Conclusões.	142
IV. - ESTRUTURAS DE DADOS E OPERAÇÕES NO MRD.	
IV.1 - Introdução.	147
IV.2 - A Estrutura de Dados Independente do Caminho de Acesso no Modelo de Representação de Objetos.	149
IV.3 - Estrutura Lógica de um SGBD apoiado no MRD.	156
IV.4 - Operações Básicas de Manipulação de Dados da Base de Dados.	164
IV.5 - Operações de Manipulação de Dados em Nível Semântico.	183
IV.6. - O Uso de uma Base de Dados MRD em Ambiente Multiusuário. . . ,	215
IV.7. - O Uso do Modelo de Representação de Objetos em uma Base de Dados para Engenharia.	223
IV.8. - Conclusões.	230

----- Parte II -----

V. - MAQUINAS ESPECIAIS PARA BASES DE DADOS.	
V.1. - Introdução.	235

V.2. - Discos Associativos.	237
V.3. - Máquinas de Base de Dados que não Restringem a Estrutura Lógica dos Dados.	242
V.4. - Máquinas de Base de Dados que Especificam a Estrutura Lógica dos Dados.	253
V.5. - Conclusões.	274
VI. - A ARQUITETURA DA MAQUINA IMPLEMENTADA.	
VI.1. - Considerações Gerais.	277
VI.2. - O Modelo de Dados do SACDS.	279
VI.3. - A Configuração da Máquina SACDS/SGBD.	282
VI.4. - A Comunicação Entre os Processadores da Máquina SACDS/SGBD.	285
VI.5. - A Memória dos Processadores da Máquina SACDS/SGBD.	295
VI.6. - Agrupamento das Informações sobre um Mesmo Objeto.	310
VI.7. - Conclusões.	313
VII. - CONSIDERAÇÕES SOBRE UMA MAQUINA DE BASE DE DADOS PARA O MODELO DE REPRESENTAÇÃO DE OBJETOS.	
VII.1. - Introdução.	315
VII.2. - A Arquitetura da Máquina de Bases de Dados Propos- ta.	318
VII.3. - A Arquitetura de um Processador de Manipulação.	324
VII.4. - O Formato de um Pacote de Dados.	328
VII.5. - A Interface do Duto de Dados.	331
VII.6. - A Identificação de Páginas que Contém as Informa- ções Pedidas.	334
VII.7. - Os Vários Tipos de Processadores de Manipulação.	339

VII.8. - Conclusões.	345
VIII. - CONCLUSÕES E LINHAS DE FUTURAS PESQUISAS.	
VIII.1. - Considerações Gerais.	349
VIII.2. - O Uso de uma Máquina de Base de Dados Específica. .	350
VIII.3. - O Uso do MRD em Aplicações que não de Projeto. . .	352
VIII.4. - Desenvolvimento de Ferramental Matemático para Análise do Modelo de Representação de Objetos. . .	355
VIII.5. - Alterações na Estrutura de Acesso Independente do Caminho de Acesso do Modelo de Representação de Objetos.	360
VIII.6 - Considerações Sobre o Significado do que é Armaze- nado pelo Modelo de Representação de Objetos. . . .	363
VIII.7 - Continuação do Aumento de Capacidade de Represen- tação Semântica do Modelo.	364
BIBLIOGRAFIA.	367

CAPÍTULO I

INTRODUÇÃO

I.1. - Considerações Gerais.

Os Sistemas de Gerenciamento de Bases de Dados (SGBD) existentes atualmente foram em sua grande maioria desenvolvidos tendo em vista aplicações comerciais, em que a tônica consiste na manipulação de um volume muito grande de dados estruturalmente idênticos. Por outro lado, aplicações não comerciais dos sistemas de computação, que vêm aumentando muito, com frequência manipulam dados que não tem aquela mesma homogeneidade estrutural. Nesse último caso encontram-se os sistemas de Projeto Apoiado por Computador - PAC (CAD), Fabricação Apoiada por Computador - FAC (CAM), Sistemas de Automação de Escritório, Inteligência Artificial, etc.

As aplicações que se desenvolvem nessas áreas, e que necessitam de Sistemas de Gerenciamento de Bases de Dados, em geral aproveitam os que já existem, mesmo que não sejam os ideais para essas aplicações em especial. Dessa forma, normalmente o emprego dos SGBDs é feito de maneira forçada, recorrendo-se a técnicas estranhas ou pouco recomendadas para o uso desses Sistemas, ou mesmo introduzindo-se elementos estranhos dentro do próprio

sistema. Assim, fica claro que essas aplicações apresentam-se como sistemas que necessitam de SGBDs, mas como aplicações não convencionais, para os quais os SGBDs atualmente disponíveis não são adequados.

Em decorrência disso, tem se tornado freqüente a sugestão da necessidade de novos SGBDs, apoiados em modelos que os tornem aptos a suprirem com eficiência e flexibilidade as necessidades de aplicações especiais. Várias propostas e algumas implementações têm surgido, porém em geral elas concentram-se nas necessidades de um espectro reduzido de aplicações.

Além disso, é muito freqüente que duas ou mais aplicações não convencionais, desenvolvidas de maneira independente, acabem tendo que ser integradas em um único sistema. Isso ocorre por exemplo quando um sistema de Projeto Apoiado por Computador deva ser integrado a um sistema de Fabricação Apoiado por Computador - PAC/FAC ou **Sistemas de Fabricação Integrada (CIM)**. Nesses casos, a falta de um núcleo de gerenciamento de dados comum traz problemas de compatibilidade entre os sistemas, sendo necessário então a criação de programas de adaptação complexos, e que em geral impedem a utilização plena das capacidades de pelo menos um dos sistemas integrados [LEA_83], [SIM_83], [WSU_86].

Sistemas de Gerenciamento de Bases de Dados que possam dar o suporte necessário a um amplo espectro de aplicações não convencionais, e que portanto possam ser empregados em sistemas que devem ser integrados em sistemas maiores, têm sido pesquisados, e o levantamento de suas necessidades tem sido apresentados, tal como em [HAR_85], [BIC_86] e [NAV_86], tendo-se tornado comum o termo **"Sistema de Gerenciamento de Dados para Engenharia"** para designá-los.

O nosso interesse por essa área vem desde 1974, quando estávamos trabalhando na implementação de uma versão simplificada do Sistema PSL/PSA (Problem Statement Language/Problem Statement Analiser) [TEI_77]), documentado em [TRA_76a] e [TRA_76b]. Esse sistema consiste em um conjunto de analisadores para uma linguagem de especificação de sistemas de software (PSL), e de um conjunto de analisadores e geradores de relatórios sobre as descrições armazenadas (PSA), que podem mostrar as mesmas informações de várias formas diferentes. Desde então passamos a pesquisar maneiras de se representar estruturas para armazenar a descrição de sistemas em computador, tal como relatado em [TRA_77], tendo desde essa época sido eleito para isso um modelo de dígrafos representados através de listas [TRA_78].

Com base nesses estudos foi projetado e implementado o Sistema de Gerenciamento de Bases de Dados usado para desenvolver o ~~Sistema de Apoio por Computador à Documentação de Sistemas~~ - SACDS, objeto de nosso trabalho de Dissertação de Mestrado [TRA_82]. Trata-se de um sistema para apoiar a tarefa de desenvolvimento de sistemas em geral, e em especial o desenvolvimento de sistemas de software. Como tal, pode ser visto como um sistema de ~~Projeto de Software Apoiado por Computador~~ - PSAC (CASE), embora possa atender ao desenvolvimento de outros tipos de sistemas.

Usando os conceitos apresentados em [TRA_82] e o próprio SACDS, vários outros trabalhos foram concebidos e estão em andamento, tal como o desenvolvimento de ferramentas para Análise de Validade, Completeza e Consistência de descrições de sistemas, e geradores de relatórios voltados para metodologias específicas

utilizando recursos gráficos.

Uma das principais características do SACDS é ser constituído de um Meta-sistema de documentação, ou seja, onde a própria linguagem e metodologia de descrição e documentação pode ser definida pelo usuário do sistema, e é dessa característica que resulta o amplo espectro de abrangência de sistemas que podem ser apoiados por esse sistema.

Porém, como tudo tem seu preço, a flexibilidade do SACDS de poder ser usado para apoiar o desenvolvimento de uma ampla gama de sistemas, com diferentes metodologias, resulta na necessidade de que o próprio usuário defina a metodologia, elabore a linguagem de descrição específica da metodologia, e estruture essa linguagem para sua introdução no SACDS. Além disso, para poder atender a uma ampla gama de linguagens e metodologias, todas as ferramentas do SACDS; seus analisadores de linguagem, seus analisadores de descrições, geradores de relatórios, etc, requisitam do usuário um volume de informações muito grande, uma vez que todas as regras da metodologia em uso devem ser passadas para o SACDS pelo próprio usuário. Tudo isso torna o SACDS um sistema que pode ser usado em muitos tipos de sistemas, mas o seu preparo para ser usado com um novo tipo de sistema ou nova metodologia é uma tarefa que requer pessoal especializado tanto na metodologia quanto nas tarefas de incorporação dessas metodologias ao SACDS, não sendo em absoluto uma tarefa fácil.

Dessa constatação, observou-se que, o que torna o SACDS realmente apto a apoiar a descrição de sistemas de muitos tipos é a sua característica de Meta-Sistema, e isso advém de sua Base de Dados, que permite que a própria definição da estrutura dessa

base provenha de uma outra Base de Dados. O SACDS pode então ser visto como um sistema que usa uma base de dados que pode dar apoio à descrição de sistemas de muitos tipos, junto com um conjunto de programas de aplicação, que constituem as ferramentas do SACDS.

Com esse enfoque, começamos a participar no Centro Tecnológico para a Informática - CTI, do desenvolvimento de um outro sistema de apoio a projetos, dessa vez especificamente para sistemas de software, denominado Sistema Integrado para a Produção de Software - SIPS [TSU_85] [TRA_86]. Trata-se de um sistema que usa uma Base de Dados apoiada em conceitos semelhantes àquela usada no próprio SACDS, com vários conceitos novos incorporados, estudados especialmente para permitir ao SIPS a representação de várias outras situações não previstas para o SACDS.

Em nossa dissertação de mestrado sugeria-se que o SACDS poderia realizar atividades automáticas pré-programadas ou pré-definidas pelo próprio usuário quando condições específicas ocorressem. Para que as atividades automáticas, e outras decorrentes da ação direta do usuário que poderiam ser desenvolvidas em paralelo, não afetassem o desempenho do sistema, sugeria-se também que o SACDS poderia ser usado em sistemas de Multiprocessadores, onde as tarefas originadas tanto pela execução do gerenciador da base de dados, quanto das próprias ferramentas, poderiam ser distribuídas. Para criar um tal ambiente apropriado, projetamos e implementamos no Instituto de Física e Química de São Carlos (IFQSC-USP), um sistema de multiprocessadores que permitisse a experimentação desses conceitos.

Assim, baseados: no desenvolvimento e utilização do SACDS; no desenvolvimento do SIPS e no estudo dos requisitos por eles impostos; no desenvolvimento e experimentação com o sistema de multiprocessadores implementado; e nas solicitações e experiências relatadas por outros grupos de pesquisa, desenvolvemos um novo modelo, denominado Modelo de Representação de Objetos - MRO, para o desenvolvimento de Sistemas de gerenciamento de Bases de Dados que devem dar apoio a sistemas integrados de projeto e produção de sistemas em geral, e além disso, que pudesse tirar proveito de arquiteturas de processamento paralelo, e em especial, que pudesse ser executado com eficiência em arquiteturas de máquinas desenvolvidas especialmente para a execução desse modelo.

A descrição desse modelo e do sistema de multiprocessadores desenvolvido para apoiá-lo, e a proposta de uma Máquina de Base de Dados especial que poderia dar suporte a bases de dados muito grandes que usem esse modelo, constituem a essência deste trabalho.

I.2. - Apresentação.

Esta tese envolve duas áreas de conhecimento, nomeadamente a de modelagem de dados para Sistemas de Gerenciamento de Bases de Dados, e a de desenvolvimento de Máquinas de Base de Dados. Esta tese é um exemplo de que ambas as áreas têm muito em comum, embora existam também muitos estudos e trabalhos que se desenvol-

vam dentro de exclusivamente uma dessas áreas, e esses constituem-se no caso mais freqüente. Assim, é comum que as atividades de modelagem de dados sejam desenvolvidas dentro de um universo próprio, que mantém poucas intersecções com as atividades de desenvolvimento de máquinas de base de dados, e vice-versa.

Devido a isso, esta tese apresenta-se também dividida em duas partes. A primeira, tem como objetivo principal a definição e o desenvolvimento de um modelo de dados, sendo constituída pelos capítulos II, III e IV. A segunda, tem como objetivo principal o estudo de máquinas especiais para base de dados, e a proposta de uma arquitetura específica que suporte esse modelo, sendo constituída pelos capítulos V, VI e VII.

No entanto, apesar do enfoque de cada uma das partes ser diferente, a unidade do trabalho como um todo se mantém, pois o desenvolvimento do modelo apresentado na primeira parte tem a preocupação de poder ser suportado por uma máquina específica que o execute, e a segunda parte procura desenvolver uma máquina que suporte o modelo apresentado na primeira.

O capítulo II tem como objetivo a identificação, apresentação e consolidação das principais necessidades e conceitos envolvidos na elaboração de um modelo de dados, adequado para o desenvolvimento de um Sistema de Gerenciamento de Bases de Dados que possa apoiar todas as fases de projeto e produção de um produto de engenharia. São levantados os principais modelos de Bases de Dados existentes atualmente, os requisitos necessários que um SGBD deve atender para dar suporte a aplicações de engenharia, sendo então feita uma comparação de quais necessi-

dades não são atendidas satisfatoriamente com aqueles modelos. É feito também um levantamento dos principais modelos semânticos existentes e propostos recentemente, e são apresentados alguns conceitos envolvendo sistemas de processamento distribuído e gerenciamento de configuração de software. Esses conceitos são apresentados e desenvolvidos, tendo por objetivo a utilização que se fará deles em capítulos subseqüentes.

No capítulo III apresenta-se o Modelo de Representação de Objetos, destinado à modelagem de Bases de Dados. Trata-se de um modelo que permite a representação de grande conteúdo semântico para os sistemas de apoio à documentação e desenvolvimento de projetos, permitindo a construção de Meta-Sistemas, em que a especificação dos requisitos da própria aplicação sejam mantidos na Base de Dados. Nesse capítulo, a apresentação do Modelo é feita de maneira gradativa, introduzindo-se os vários conceitos progressivamente, tomando-se como ponto de partida o Modelo Entidade-Relacionamento. É feita a seguir uma discussão de vantagens decorrentes do uso desse modelo em ambientes de multiprocessamento.

No capítulo IV, o Modelo de Representação de Objetos é discutido em termos de seus componentes, maneira de implementação e características semânticas. É feita uma apresentação das estruturas lógicas de acesso que permitem a implementação de um Sistema de Gerenciamento de Base de Dados (SGBD) usando esse modelo, e de um conjunto mínimo de operações que devem ser oferecidas por um tal sistema. Essas operações são apresentadas em três níveis de progressiva expressividade semântica, que permitem ao SGBD abstrair progressivamente mais significado das operações que são solicitadas pelos programas de aplicação. Ao final desse

capítulo mostra-se como os requisitos das aplicações de sistemas de gerenciamento de dados para projeto e engenharia são atendidos pelo modelo.

No capítulo V é feito um levantamento bibliográfico de máquinas de bases de dados já projetadas, buscando-se identificar as principais características e conceitos desenvolvidos, que possam ser utilizados no desenvolvimento de uma máquina que suporte o Modelo de Representação de Objetos com eficiência. Dessa forma, esse capítulo apresenta um panorama das atividades em andamento nesse campo, ao mesmo tempo em que se traça uma linha da história de desenvolvimento dessas máquinas.

No capítulo VI é feita a apresentação de uma máquina que utiliza múltiplos micro-computadores fortemente acoplados, projetada e implementada para permitir a execução do SACDS de maneira que as suas ferramentas e o SGBD que ele utiliza pudessem ser executadas em paralelo em processadores separados. Apresenta-se a estrutura geral dessa máquina, e analisam-se com maior profundidade os conceitos e soluções mais importantes empregados, tal como a estrutura de memória de cada processador, a forma de comunicação adotada entre eles, e a maneira de se estruturar os dados dos vários arquivos para se conseguir um aumento na eficiência de sua execução.

Como resultado do desenvolvimento dessa máquina, foram identificados vários parâmetros que puderam ser usados para que se estabelecessem as especificações de uma Máquina de Bases de Dados, que poderia suportar com eficiência o Modelo de Representação de Objetos. No capítulo VII são feitas considerações sobre esses parâmetros e se apresenta a sugestão de uma máquina de base de dados específica, com características que a torna indicada

para a execução de um SGBD que use o Modelo de Representação de Objetos, e é feita uma discussão sumária de como isso pode ser realizado.

No capítulo VIII são feitas algumas considerações, e são apresentadas conclusões sobre o uso do Modelo de Representação de Objetos em algumas aplicações de projeto e de manipulação de dados de engenharia específicas, e sobre a implementação de um SGBD que o use, bem como sobre a implementação da Máquina proposta no capítulo V. Apresentam-se também várias sugestões sobre tópicos que devem ser estudados para continuar-se a pesquisa e o desenvolvimento dos assuntos tratados nesta tese.

CAPÍTULO II

APRESENTAÇÃO DO PROBLEMA

II.1. - Introdução.

Cada vez mais a informática alcança novos horizontes, cada vez é maior o número de áreas da atividade humana que se automatizam, cada vez é maior o número de aplicações em que o computador passa a tomar parte. E conforme aumenta o número de aplicações que se utilizam do computador em uma organização, maior é a necessidade de integração das informações e dos sistemas necessários para que o computador possa desempenhar as novas funções que lhe vão sendo atribuídas.

Os sistemas computadorizados que constituem essas aplicações dependem fundamentalmente dos dados com que são alimentados, e pode-se dizer que a própria essência da existência desses sistemas é a informação que eles manipulam, e à qual dão saída na forma de dados para outros sistemas também computadorizados, para as próprias pessoas, ou controlando outros dispositivos, para os quais são com frequência especialmente dedicados.

Desde os primórdios do surgimento do computador, as tarefas de manipulação, armazenamento e recuperação de dados têm sido estudadas, e hoje existem sistemas que podem satisfazer grande

parte dessas necessidades, nos empregos mais tradicionais do computador como uma ferramenta de "Processamento de Dados", fazendo surgir os conhecidos Sistemas de Gerenciamento de Bases de Dados.

No entanto, para as novas aplicações em que o computador está sendo empregado, esses sistemas com frequência apresentam severas restrições de uso, levando à procura de soluções inovadoras que atendam às novas necessidades. O projeto de sistemas que possam tornar disponíveis os dados necessários para uma dada aplicação conforme requisitado, numa forma conveniente, é o objeto de estudo de uma disciplina a que se convencionou chamar Engenharia de Dados.

Este trabalho tem como objetivo identificar as necessidades de manipulação, armazenamento e recuperação de dados em aplicações ligadas à engenharia, ou seja: aplicações que envolvem o emprego de conhecimento e técnicas bem determinadas para criação de estruturas, dispositivos ou processos, que são utilizados para converter recursos existentes em formas mais adequadas para o atendimento das necessidades humanas. Em outras palavras, este trabalho procura tratar do emprego da Engenharia de Dados a sistemas de apoio à engenharia em todas as suas manifestações.

Esse trabalho envolve uma grande quantidade de áreas, pois as tarefas associadas ao projeto e produção de bens, oriundos das diversas especializações de engenharia, envolvem os requisitos necessários a praticamente todas as atividades da engenharia. Assim, como será visto neste capítulo, as necessidades de uma Base de Dados para Engenharia engloba as necessidades de aplicações específicas de áreas tais como Projeto Auxiliado por Computador, Fabricação Auxiliada por Computador, Automação de

Escritórios, Inteligência Artificial, Coleta e Processamento de Dados para Experimentação Científica, etc. Dessa forma, pode-se considerar que uma Base de Dados para Engenharia compreende todos os recursos necessários para a integração de uma grande quantidade de aplicações hoje consideradas não tradicionais para as bases de dados, além dos recursos tradicionais, já amplamente estudados e disponíveis.

Na seção II.2 deste capítulo, apresentam-se sumariamente os conceitos e os modelos de bases de dados mais comuns existentes, e na seção II.3 as necessidades que caracterizam as Bases de Dados para engenharia. Na seção II.4 são apresentados alguns modelos de dados não tradicionais, que têm sido sugeridos na literatura como capazes de satisfazer necessidades de determinadas aplicações não convencionais de Bases de Dados. A seguir, na seção II.6 são estudados alguns tópicos que podem auxiliar o desenvolvimento de um modelo de base de dados indicado para aplicações de engenharia, tais como o processamento paralelo de dados, a consistência e as maneiras de se manter a integridade de uma Base de Dados, e o controle de versões durante o desenvolvimento de um projeto.

II.2 - Os Principais Modelos de Bases de Dados Usados Atualmente.

Os Sistemas de Gerenciamento de Bases de Dados (SGBD) surgiram da necessidade de melhorar os sistemas de Processamento de Dados existentes no início dos anos 60. Essa melhora era desejada para resolver fundamentalmente dois problemas: o

problema de independência de dados; e o problema de redundância de dados. Leve-se em conta, que na época em que tais necessidades surgiram, os sistemas de processamento tinham que se haver diretamente com os arquivos de dados das aplicações das quais tratavam, e o surgimento dos SGBDs vieram em resposta principalmente para, sistematizando o acesso aos arquivos de dados, dar uma solução a esses dois problemas [BRA_81].

O problema de independência de dados diz respeito à necessidade de uma aplicação usar os dados que precisa sem que tenha de levar em conta a estrutura física que armazena esses dados. Dessa forma, uma aplicação não precisa ser alterada se a estrutura dos dados mudar. Um SGBD oferece para o programa de aplicação, operações para a manipulação dos dados que armazena, ao mesmo tempo em que impede o programa de fazer qualquer referência à estrutura real usada para armazenar esses dados.

O problema de redundância de dados advém da possibilidade de se armazenar o mesmo dado em mais de um lugar, ou em mais de uma estrutura ao mesmo tempo. Essa repetição de informações deve ser evitada, pois pode trazer problemas sérios de atualizações, ou de conflito de informações. Um SGBD armazena todos os dados que fazem parte de uma Base de Dados sem permitir repetições, ao mesmo tempo que permite a visão desses dados sob diferentes pontos de vista, permitindo que cada programa de aplicação use os dados da forma como necessita, sempre se referindo a eles de uma maneira que lhe seja conveniente.

Para evitar esses dois problemas, um Sistema de Gerenciamento de Bases de Dados usa três esquemas diferentes de representação de dados, caracterizando três níveis de abstração (Figura 1): o ~~esquema físico~~ da base de dados, caracterizando a

forma como a base de dados armazena efetivamente os dados; o esquema conceitual da base de dados, caracterizando como os dados são vistos como um todo, de forma lógica, pelo indivíduo que atua como gerente da base de dados; e o conjunto de esquemas dos vários programas de aplicação, comumente chamados de "Visão de Dados" de um programa de aplicação, caracterizando como cada programa "vê" os dados armazenados na Base de Dados.

Figura 1 - Os níveis de abstração de uma Base de Dados.

O esquema físico de uma base de dados não diz respeito à nenhum programa de aplicação, sendo uma característica intrínseca a uma dada implementação de SGBD. Algumas vezes é desmembrado em duas especificações de estrutura de dados: **Estrutura de Dados Dependente do Caminho de Acesso** à informação, que diz respeito à estrutura de acesso efetivamente usada para estruturar a informação, como por exemplo onde e como se usam estruturas de

árvores, "hash", listas, etc.; Estrutura de Dados Independente do Caminho de Acesso à informação, que diz respeito a forma como os dados são estruturados, constituindo arquivos e estruturas, sem que sejam necessários detalhes da implementação física das estruturas dos ponteiros que são agregados à Informação que a Base de Dados armazena.

Todos os dados armazenados em uma Base de Dados tem uma estrutura lógica, que representa a maneira abstrata com que uma pessoa entende os inter-relacionamentos existentes entre os dados armazenados, em termos de como os dados são interdependentes, representando conceitos de hierarquia, continência, identidade, etc.

As várias Visões de Dados nada mais são do que a forma como cada aplicação percebe a estrutura lógica dos dados que manipula. Os dados de uma aplicação em geral são parte dos dados armazenados na Base de Dados. O esquema conceitual é o esquema lógico de todos os dados armazenados na base de dados, ao passo que cada Visão de Dados corresponde a um pedaço do todo, permitindo a cada programa de aplicação, o acesso somente aos dados que ele precisa, ou que está autorizado a manipular.

Para efeito dos acessos que cada programa realiza na base de dados, considera-se que cada programa é um programa de aplicação, às vezes chamado apenas de "Aplicação", que pode, em uma base de dados compartilhada, ser executado em paralelo com outro.

Cada aplicação é composta de várias operações, denominadas "Ações na Base de Dados", como por exemplo Leitura ou Escrita de dados. Essas ações devem ser efetuadas de tal maneira que mantenham a consistência da base de dados.

Porém, pode ocorrer que, entre duas ações simultâneas, a

consistência seja temporariamente corrompida [ESW_76]. Por exemplo, quando efetua-se a inserção ou retirada de alguma informação em uma tabela, para a qual se mantém o total de uma coluna também armazenada, entre a inserção ou retirada do novo dado (que corresponde a uma ação) e a atualização do total (outra ação), existe um intervalo de tempo em que a informação não estava consistente. As ações são então logicamente agrupadas em Transações, de tal forma que cada transação sempre mantenha a consistência da base. No exemplo acima, as ações de inserir um novo dado e atualizar o total juntas poderiam ser uma transação. Assim, uma aplicação é composta por transações, que garantem a integridade da Base de Dados, e estas por sua vez são compostas de ações.

Um outro conceito importante quando trata-se de Bases de Dados, é que as informações podem ser acessadas tanto através de uma referência direta, através da especificação de um identificador, quanto através de referências indiretas, através dos relacionamentos que os dados mantêm com outros. Essa segunda forma de acesso, pressupõe a existência de um ponteiro indicando o dado sendo acessado, e através do deslocamento desse ponteiro para outros dados que se relacionam com a posição anterior, pode-se acessar esses outros dados, mesmo sem uma identificação explícita. Essa operação é denominada "Navegação da Base de Dados".

Existem várias maneiras de realizar-se uma implementação de um Sistema de Gerenciamento de Bases de Dados, e cada implementação sempre se atém a uma particular maneira de se considerar a passagem da representação de dados do mundo real para o domínio da representação da base de dados. O domínio de representação de

dados de uma Base de Dados é chamado Modelo de Dados ou Modelo de uma Base de Dados.

Atualmente existem quatro Modelos de Bases de Dados comumente usados nas várias implementações existentes: Modelo de Rede; Modelo Hierárquico; Modelo Relacional; e Modelo Entidade-Relacionamento. Nas sub-seções seguintes cada um deles será sumariamente apresentado.

O modelo de uma Base de Dados é caracterizado pelo esquema conceitual da base de dados, e indica aproximadamente como deve ser a Estrutura de Dados Independente do Caminho de Acesso do esquema físico da base de dados. Dessa forma, o modelo de dados escolhido para uma Base de Dados faz com que o SGBD possa ser empregado convenientemente para uma dada aplicação, e não para outra. Além disso, o Modelo de Dados estabelece a Linguagem de Definição de Dados que é usada para especificar a estrutura dos dados de uma aplicação, e atualmente existe para cada um desses modelos mais comuns até terminologia e simbologia próprias.

II.2.1 - O Modelo de Rede.

O modelo de Rede foi o primeiro a ser tratado em ampla escala, e a ter alguns padrões estabelecido, através de um grupo de trabalho envolvendo usuários e fabricantes, conhecido como CODASYL ("Conference on Data Systems Language"), cujo objetivo original era especificar uma linguagem universal para as necessidades comerciais, da qual nasceu a especificação da linguagem COBOL. Conforme o interesse em bases de dados cresceu,

esse grupo criou em 1965 um comitê especial para cuidar desse assunto, denominado DBTG (Data Base Task Group). Desse grupo resultou a especificação de um Sistema de Gerenciamento de Bases de Dados, hoje conhecido como esquema conceitual da CODASYL, e tão amplamente conhecido, que frequentemente toma-se o termo Modelo CODASYL como sinônimo de Modelo de Rede [BRA_81].

Nesse modelo, parte-se do conceito de que as informações são armazenadas em arquivos, onde os dados com estrutura idêntica são agrupados. Cada arquivo é composto de um conjunto de registros, todos com a mesma estrutura (mesmo formato). A definição de um registro, chamada simplesmente Registro, estabelece o conteúdo de um arquivo associado, e cada registro desse arquivo é chamado uma Instância de Registro.

Cada registro é composto de Campos, e cada ocorrência de um campo em uma instância de um registro é chamada de Instância de um Campo. Os vários campos que compõem um registro são justapostos, e podem conter informação, quando são então chamados de Campos Primários, ou podem se referir à informações que são armazenadas em outros registros, quando então são chamados de Campos de Referência.

Podem existir relacionamentos de cardinalidade 1:N ou de 1:1. Os relacionamentos de cardinalidade N:M não são diretamente representados no modelo de rede, sendo desmembrados em dois, um de cardinalidade 1:N e outro de cardinalidade 1:M, entre os dois registros envolvidos.

O registro que ocorre apenas uma vez no relacionamento de cardinalidade 1:N é chamado registro Mestre ("Owner"), e o registro que pode ter múltiplas ocorrências é chamado registro Membro ("Member"). No relacionamento de cardinalidade 1:1, é considerado

mestre o registro que tiver um potencial de identificação maior (o registro que nos programas de aplicação se espera ter maior número de acessos através da identificação de um campo chave).

Relacionamentos de cardinalidade 1:1 são pouco usados, pois em geral representam extensões do registro mestre, e sempre existe para cada Registro Mestre um Registro Membro. Devido a isso, relacionamentos de cardinalidade 1:1 normalmente são utilizados apenas em visões de programas de aplicação.

No Modelo de Rede, cada relacionamento é denominado Conjunto ("Set"), e um relacionamento junto com todas as ocorrências de registros mestre e registros membro são denominadas Ocorrência de Conjunto, ("Occurrence Set"), constituindo-se no principal meio de acesso de informações na base de dados.

Com o Modelo de Rede, normalmente usa-se a notação de Bachman [BAC_69] para expressar simbolicamente a estrutura de dados de uma dada aplicação. Na figura 2 está um exemplo de um modelo de dados representado nessa notação, representando um sistema de arquivamento bibliográfico.

Figura 2 - Diagrama de Bachman Estendido.

Em um diagrama de Bachman, são representados todos os registros e os campos que os constituem, sendo que os campos usados como chaves de identificação são grifados. Os conjuntos que existem são identificados através de setas apontando do registro Mestre para o registro Membro.

Os conjuntos expressos pelos diagramas de Bachman em geral mostram relacionamentos de cardinalidade 1:N, embora relacionamentos de cardinalidade 1:1 também possam ocorrer. Devido ao muito maior número de casos de relacionamentos de cardinalidade 1:N em comparação ao de cardinalidade 1:1, em um diagrama de Bachman costuma-se omitir a indicação cardinalidade 1:1, explicitando apenas as de cardinalidade 1:N.

No exemplo da figura 2, a base de dados estaria apoiada em um esquema conceitual consistindo de três arquivos: Referência, Autores, e Editora. Uma referência seria composta de um Título, número de páginas, um ou mais autores, e uma editora que o

publica. Um autor teria armazenado um nome que o identifica, um endereço, e teria um ou mais trabalhos publicados. Por fim uma editora teria um nome e um endereço.

II.2.2 - O Modelo Hierárquico

Em Sistemas de Gerenciamento de Bases de Dados que usam este modelo, os registros podem ser agrupados de duas maneiras. Inicialmente, é feito o agrupamento dos registros em arquivos, sendo que todos os registros de cada arquivo devem ser do mesmo formato, e a seguir agrupam-se os registros em tipos hierárquicos, correspondendo à hierarquia de ocorrências, ou de identificação e acesso dos registros.

Nesse modelo, os registros mantêm entre si relacionamentos semelhantes aos que ocorrem no modelo de Rede, porém o registro que origina um relacionamento (aquele do qual existe apenas uma ocorrência) é chamado de Registro Pai, e o outro registro do relacionamento (que pode ocorrer múltiplas vezes) é chamado de Registro Filho.

Em um Modelo Hierárquico existe a restrição de que um registro pai pode estar relacionado com um ou muitos registros filhos (um pai pode ter muitos filhos), porém um registro filho pode estar relacionado a apenas um registro pai. Na Figura 3 está esboçado um diagrama de Bachman genérico para o Modelo Hierárquico.

Figura 3 - Modelo Hierárquico genérico

A forma de implementação física do modelo Hierárquico está bastante ligada ao esquema conceitual, uma vez que as duas formas de agrupamento que esse modelo apresenta se referem à forma como essas estruturas são fisicamente mantidas. Isso se deve à necessidade de se representar através do modelo hierárquico, situações do mundo real que nem sempre estão estruturadas de maneira hierárquica. Para isso, o modelo prevê a existência de representação de dois modelos conceituais simultâneos para um mesmo problema do mundo real, que se denominam Hierarquia Primária e Hierarquia Secundária.

O modelo conceitual que representa a Hierarquia Primária, é usado pelo sistema de Gerenciamento da base de dados para determinar a estrutura física da base de dados, e é dessa forma que o acesso aos dados é efetuado. A Hierarquia Secundária, que em algumas implementações pode até constituir um conjunto de várias hierarquias, fornece caminhos de acesso lógicos alterna-

tivos, que podem ser usados para a interface entre o SGBD e o programa de aplicação.

Dessa forma, o modelo hierárquico não está restrito a aplicações que tenham uma estrutura hierárquica, pois usando-se modelos conceituais que envolvem a representação de Hierarquias secundárias, este pode ser usado em aplicações que envolvam um limitado número de características de rede. No entanto, deve-se levar em conta que, a menos dos acessos efetuados através da hierarquia primária, os acessos são mais custosos em termos de tempo de acesso, e devem ser usados com parcimônia.

II.2.3. - O Modelo Relacional.

O Modelo Relacional tem seus fundamentos originados na teoria matemática da álgebra das relações, tendo sido desenvolvido a partir dos trabalhos de Codd [COD_70], numa tentativa de desenvolver um modelo para um SGBD que apresentasse para o projetista de aplicação uma interface de mais alto nível do que as apresentadas pelas interfaces dos SGBD existentes na época. O conceito matemático da álgebra das relações é a operação do produto cartesiano de dois ou mais conjuntos:

$$A \times B \times C \times \dots$$

do qual escolhem-se as n-uplas ordenadas que constituem cada relação.

O modelo relacional foi estudado durante muito tempo, sendo que as primeiras implementações realizadas foram feitas com o propósito de estudar o modelo, ou criar ambientes de estudos, onde as propostas subsequentes pudessem ser experimentadas. Nesse sentido foram bastante divulgadas as implementações do projeto "System R" [AST_76] desenvolvido pela IBM, e o projeto INGRES (Interactive Graphics and Retrieval System) [STD_76] desenvolvido na Universidade da Califórnia em Berkeley. Desses sistemas, o System R tem servido como um modelo para a implementação de sistemas comerciais, atuando como uma espécie de referência, para as implementações atuais.

O modelo relacional pode ser apresentado [DAT_75], considerando que as informações são agrupadas em tabelas, denominadas relações, cada uma agrupando informações que mantêm a mesma estrutura (registros, segundo a denominação adotada nos modelos de rede e hierárquico), e que na terminologia adotada pelo modelo relacional são chamadas de tuplas. Correspondendo ao conceito matemático de que uma relação é um conjunto, vem o fato de que cada tupla em uma relação é única, não podendo existir em uma relação mais do que uma tupla contendo os mesmos valores.

O grau de uma relação corresponde ao número de colunas na relação, quando se considera a relação como uma tabela. Por exemplo, uma tabela que tenha 3 colunas corresponde a uma relação de grau 3, e cada tupla dessa relação é composta de 3 informações. Cada uma das colunas tem um nome, que é denominado atributo da relação na terminologia própria do modelo relacional. Ao conceito de Atributo corresponde o conceito de campo dos modelos de rede ou hierárquico (não à instância, mas à definição do campo).

O conjunto de valores que um atributo pode assumir é definido como o domínio de um atributo. No modelo relacional estende-se esse conceito, estabelecendo-se o conceito de domínio de uma relação, que consiste nos conjuntos de valores que cada atributo da relação pode assumir.

Um atributo ou um conjunto de atributos de uma relação deve poder identificar univocamente cada tupla da relação. Um conjunto mínimo de atributos que possa ser usado para identificar cada tupla é denominado chave candidata. Desde que mais de um conjunto de atributos podem ser usados como identificadores, uma mesma relação pode ter mais de uma chave candidata. Um desses conjuntos que tenham o menor número possível de atributos envolvidos será escolhido para constituir a chave de identificação primária da relação.

A operação de um Sistema de Gerenciamento de Base de Dados Relacional fundamenta-se no conceito de normalização de relações, um conceito introduzido no trabalho [COD_70], e que hoje estende-se a todos os modelos de bases de dados. A normalização de relações consiste em criar-se relações que não tenham atributos cujos valores dependam de outros atributos já existentes na mesma relação. Por exemplo, na figura 4 aparece uma relação de ordens de compra, contendo como atributos: Número da ordem, Data da Compra, Nome do Fornecedor, Número do Fornecedor, Número do Produto, Quantidade do Produto.

Relação de Ordens de Compra

Número Ordem	Data	Nome do Fornecedor	Número do Fornecedor	Número do Produto	Quanti- dade
00123	23/05/85	Porcelanas Cristal	19	77-860	1
00123	23/05/85	Porcelanas Cristal	19	77-241	10
00145	21/06/85	Porcelanas Cristal	19	75-305	10
00205	10/10/85	Fadinha Ind. Com.	06	52-021	100
00205	10/10/85	Fadinha Ind. Com.	06	50-103	50

Figura 4 - Relação não normalizada.

Essa relação não está normalizada, pois existem atributos cujos valores podem ser obtidos diretamente através de outros atributos da mesma relação, como é o caso do Nome do Fornecedor, que pode ser obtido do Número do Fornecedor. Essa situação não é desejável, uma vez que a duplicação de informações pode levar a inconsistência de dados. A figura 5 apresenta uma versão normalizada dessas mesmas informações.

O processo de normalização geralmente leva à criação de mais relações, tal como ocorre no exemplo, em que uma relação é separada em três, porém assegura uma maior consistência de dados, bem como, em geral, diminui a quantidade de espaço necessário para armazenar as informações. Existem regras bem estabelecidas para se proceder à normalização, que podem por exemplo ser encontradas em [GAN_79].

Relação de Produtos por Ordem de Compra

Número Ordem	Número do Produto	Quantidade
00123	77-860	1
00123	77-241	10
00145	75-305	10
00205	52-021	100
00205	50-103	50

Relação de Ordens de Compra

Número Ordem	Data	Número do Fornecedor
00123	23/05/85	19
00145	21/06/85	19
00205	10/10/85	06

Relação de Fornecedores

Número do Fornecedor	Nome do Fornecedor
06	Fadinha Ind. Com.
19	Porcelanas Cristal

Figura 5 - Relações Normalizadas.

A modelagem de um sistema usando o Modelo Relacional é feita através da especificação de cada relação que a Base de Dados deva armazenar. A representação de um esquema conceitual, quando se usa esse modelo, normalmente é feita através da especificação das relações segundo a notação usualmente empregada na teoria matemática da álgebra das relações. Por exemplo, as relações da Figura 5 poderiam ser especificadas da seguinte forma, na descrição de um esquema conceitual:

Produtos (NOrdem, NProduto, Quantidade)

OrdemCompra (NOrdem, Data, NFornecedor)

Fornecedor (NFornecedor, NomeFornecedor)

Note-se que a notação de Bachman pode ser usada como um recurso gráfico para a representação de esquemas conceituais também do Modelo Relacional, no entanto essa prática normalmente não é empregada.

Uma implementação de SGBD usando o modelo relacional, apresenta para o projetista de aplicações, operações de alto nível [BRA_81] [KRU_83], fundamentadas nas operações da teoria dos conjuntos, tais como seleção, projeção, união natural e divisão. Todas essas operações têm como resultado a criação de outras relações, com atributos e tuplas obtidos a partir das relações usadas como operando. Uma Base de Dados relacional armazena as relações descritas no modelo conceitual da base, e as relações que são criadas por operações são mantidas em uma área de trabalho própria de cada aplicação, e somente tem existência durante o tempo em que a aplicação está sendo executada.

Um conceito que é importante notar aqui, é que as relações não pressupõem a existência de uma ordem entre as tuplas, apesar de que na prática, as relações mantidas por um SGBD usualmente são mantidas segundo alguma ordem nas tuplas, dependendo dos valores de alguns de seus atributos.

II.2.4. - O Modelo Entidade-Relacionamento.

O Modelo Entidade-Relacionamento (ME-R) foi proposto em [CHE_76] como uma extensão ao Modelo Relacional, com o propósito de trazer para o modelo mais informação sobre a estrutura dos

dados representados, ou seja, com o propósito de aumentar o conteúdo semântico do modelo.

Esse modelo parte do pressuposto que um sistema do mundo real pode ser descrito através da identificação das "Entidades" do sistema, as quais são descritas através de suas "Propriedades", e através de "Relacionamentos" entre essas entidades. Um relacionamento pode opcionalmente ter também propriedades próprias.

Propriedades são atributos aos quais são dados valores (propriedade é um "Atributo" com um "Valor de Atributo" determinado). Por exemplo, em um sistema de referências bibliográficas, um autor será considerado uma entidade, sendo que seu nome é um atributo; um nome específico, seja por exemplo José da Silva, é um valor de atributo, e esse nome é uma propriedade do autor José da Silva.

As entidades são agrupadas em arquivos, sendo que cada arquivo armazena as entidades que tenham os mesmos atributos, constituindo os "Conjuntos de Entidades".

Da mesma maneira, agrupam-se os relacionamentos que tenham atributos iguais e que inter-relacionem entidades dos mesmos conjuntos de entidades. Dessa forma, cada relacionamento é constituído das identificações das entidades que ele inter-relaciona, e dos atributos que o relacionamento possua.

Resumindo, pode-se dizer que o Modelo Entidade-Relacionamento descreve um sistema através da representação de suas entidades, através dos relacionamentos existentes entre as entidades; e dos atributos que especificam as entidades e relacionamentos modelados. Além disso, as entidades e relacionamentos que tenham atributos iguais são agrupados, constituindo os conjuntos de entidades e conjuntos de relacionamentos.

Cada entidade em um conjunto de entidades deve ser identificada univocamente através de um conjunto de atributos. O conjunto desses atributos é denominado "Chave de Identificação" de um conjunto de entidades.

De maneira diferente do Modelo Relacional, o Modelo Entidade-Relacionamento determina que os relacionamentos existentes entre as entidades sejam especificados. Os relacionamentos podem ser de qualquer cardinalidade, ou seja, existem relacionamentos de cardinalidade 1:1, 1:N e N:M.

O esquema conceitual de uma base de dados que emprega o Modelo Entidade-Relacionamento, é feito descrevendo-se os conjuntos de entidades e os conjuntos de relacionamentos que representam o sistema que deve ser armazenado, e indicando quais são os atributos que cada conjunto possui.

O esquema de uma modelagem usando o ME-R é normalmente representado através de um diagrama, denominado Diagrama de Entidades-Relacionamento (DE-R), onde usa-se um retângulo para representar as entidades, um losango para representar os relacionamentos, e elipses para representar os atributos [CHE_78]. A Figura 6 mostra como o esquema do Sistema de Arquivamento Bibliográfico da Figura 2 ficaria nessa representação.

Apesar do Modelo Entidade-Relacionamento ter sido proposto juntamente com um conjunto de conceitos e uma terminologia próprios, deve-se ressaltar aqui que a principal diferença entre ele e o Modelo Relacional é a possibilidade que o ME-R apresenta de representar explicitamente os relacionamentos. Note-se que no ME-R, os conjuntos de entidades e os conjuntos de relacionamentos são armazenados como relações de entidades e relações de relacionamentos.

Figura 6 - Representação no ME-R.

Estabelecendo os relacionamentos explicitamente, aumenta-se a capacidade de representação do modelo, e torna-se exeqüível a recuperação de informações através da navegação dos relacionamentos da base de dados. Além disso, podendo armazenar dados particulares de cada relacionamento, aumenta-se o conteúdo semântico do modelo. O Modelo Relacional poderia armazenar os mesmos dados, porém como relações indistinguíveis das relações que armazenariam as entidades, eliminando-se a semântica que existe no fato de um dado representar um relacionamento. Por exemplo, um relacionamento pode ter cardinalidade, funcionalidade, associatividade, etc. que o ME-R captura, porém o Modelo Relacional não.

Devido ao conteúdo semântico apresentado pelo Modelo Entidade-Relacionamento ser substancialmente maior do que os apresentados pelos demais modelos, tem se tornado prática comum o uso do Diagrama de Entidades-Relacionamento para a modelagem dos dados de uma aplicação, e posteriormente efetuar-se a tradução dessa representação para diagramas voltados a uma particular implementação do SGBD que se pretende usar e para a Linguagem de Descrição de Dados daquele SGBD.

Essa prática tem feito surgir extensões ao ME-R original, no sentido de representar no diagrama maior significado dos dados do que o suportado pelas implementações onde o modelo será empregado. Assim, é comum o emprego de indicadores de exclusão mútua entre relacionamentos, indicação de que determinados atributos são não repetitivos, etc. No entanto, não existe nenhum consenso quanto a uma possível padronização dessa representação, e assim, embora o DE-R tenha se tornado um pólo para a convergência de técnicas de modelagem, representação e análise de dados, não existe um formato ou uma técnica consagrada para a representação lógica de esquemas de dados em geral.

II.2.5. - Motivação do Interesse nos Modelos Tradicionais.

Os quatro modelos apresentados constituem juntos a grande maioria dos Sistemas de Gerenciamento de Bases de Dados existentes. Entretanto o reconhecimento generalizado de que muitas aplicações em que são usados não são satisfatoriamente atendidas

por nenhum deles, tem feito surgir várias propostas para o desenvolvimento de novos modelos, mais adaptados para essas aplicações.

Este trabalho tem por objetivo a representação de informações referentes à tarefa de projeto, desenvolvimento e produção de sistemas, e portanto busca uma forma adequada de armazenagem dessas informações, através de um modelo de dados adequado para isso. Para que esse modelo possa ser concebido, é necessário o levantamento das necessidades impostas pelas aplicações que o utilizarão. Isso é feito na seção seguinte, analisando-se essas necessidades através do reconhecimento dos vários aspectos que se apresentam em sistemas voltados para o projeto, desenvolvimento, comunicação de dados obtidos, e produção de sistemas em geral.

Além dos quatro modelos já apresentados, vários outros têm sido propostos e desenvolvidos, alguns tendo mesmo o objetivo de representação de dados de projetos. Os modelos que podem suprir ao menos algumas das necessidades levantadas são apresentados na seção II.4. O objetivo dessa seção é então o de fazer um levantamento dos modelos já existentes que poderiam suprir as necessidades encontradas, ou fornecer subsídios para o desenvolvimento de um outro modelo que as supra. Os modelos tradicionais, apresentados nesta seção, apesar de não serem adequados para as aplicações de projeto, desenvolvimento e produção, têm sido amplamente empregados nessas mesmas aplicações, principalmente devido à sua alta disponibilidade.

Conforme será verificado pelo levantamento das necessidades dessas aplicações, nenhum modelo existente pode satisfazer a todas elas, sejam os modelos tradicionais ou os modelos que serão

apresentados na seção II.4. Porém, têm-se tornado patente que, genericamente, quanto maior a capacidade de representação de maior significado (maior conteúdo semântico) de um modelo, tanto maior é a capacidade de atender as necessidades de projeto e desenvolvimento de sistemas que esse modelo apresenta. Assim, pode-se entender a apresentação dos vários modelos nesta seção e na seção II.4 como uma seqüência de modelos que permitem a representação de um conteúdo semântico cada vez maior, e que apontam na direção do modelo que será apresentado no capítulo III, como produto dos trabalhos por nós realizados.

II.3. - Especificação das Necessidades das Bases de Dados para Engenharia

Na seção anterior foram apresentados brevemente os modelos mais comuns usados nos Sistemas de Gerenciamento de Bases de Dados atuais. Verificando-se a seqüência em que os vários modelos surgiram, e comparando-se a quantidade de informação sobre os dados armazenados que cada modelo incorpora, percebe-se que a tendência, e a própria justificativa do surgimento de cada novo modelo, é a maior capacidade de absorver as informações sobre os dados que cada novo modelo apresenta sobre os anteriores.

Essas informações sobre os dados armazenados, freqüentemente aludidas como dados sobre os dados, ou Meta-dados, constituem em última análise o significado, ou semântica, dos dados armazenados. Dessa forma, diz-se que a tendência dos modelos de

Bases de Dados é o contínuo aumento no seu conteúdo semântico. Por exemplo, se juntamente com os dados de um sistema forem indicados também o meio como esses dados foram obtidos, tal como qual instrumento foi utilizado para se conseguir uma determinada medida, teremos além dos dados - números que constituem a medida - outras informações, que são Meta-dados sobre os dados armazenados. Uma base de dados que possa armazenar essas informações terá um conteúdo semântico maior do que uma que não o possa.

A apresentação dos modelos na seção anterior foi feita na ordem cronológica de seu surgimento, e que corresponde à ordem crescente de seu conteúdo semântico. Assim, o Modelo Relacional, proposto e desenvolvido no início do anos 70, tem um conteúdo semântico maior do que os Modelo de Rede e Modelo Hierárquico, que surgiram uma década antes. Da mesma forma, o Modelo Entidade-Relacionamento, proposto e desenvolvido no final dos anos 70 e início dos anos 80, apresenta um conteúdo semântico ainda maior do que o apresentado pelo Modelo Relacional.

Hartzband e Maryanski [HAR_85] apresentam uma classificação dos modelos de dados segundo a quantidade de significado representado em cada modelo. Segundo essa classificação, pode-se perceber que o conteúdo semântico de um modelo aumenta conforme aumenta a variedade e a complexidade de inter-relacionamentos que podem ser representados entre as informações elementares armazenadas. Dessa forma, quanto menor for a imposição de uma estrutura pré-definida para a modelagem de uma base de dados, maior a sua capacidade de representação semântica. A ausência de uma estrutura pré-definida implica que uma estrutura deve ser definida pelo usuário para cada aplicação em que o sistema de Base de Dados seja usado, e o objetivo então passa a ser o de fornecer ao

usuário um conjunto o mais amplo possível de recursos que lhe permitam expressar como cada item de dados de interesse se inter-relaciona com os demais. Aumentando a quantidade de informações de como os dados se inter-relacionam, ou seja de Meta-dados estruturais, surge a necessidade de estruturas próprias no Sistema de Gerenciamento de Bases de Dados para a sua representação, o que indica a necessidade de uma (Meta) base de dados para a representação dos Meta-dados de uma aplicação.

Um problema associado com representação dos Meta-dados, e conseqüentemente, com a representação da semântica dos dados, é que estes constituem-se de dados que não guardam entre si uma estrutura clara, que possa ser facilmente representada pelos modelos adotados nos Sistemas de Gerenciamento de Bases de Dados atuais. Na realidade, os modelos adotados até agora quase que somente permitem a representação dos dados em formato de tabelas, o que limita enormemente a quantidade de dados não estruturados, e portanto a quantidade de Meta-dados que podem ser representados em cada aplicação.

O propósito final deste trabalho é identificar as necessidades de um sistema de gerenciamento de dados para engenharia, e propor uma forma adequada que um sistema possa empregar para suprir essas necessidades. Nas subseções seguintes é feita uma apresentação do que se convencionou chamar de uma base de dados para engenharia, bem como se apresentam as características desejáveis e/ou necessárias para uma base de dados desse tipo. Do levantamento das necessidades, constata-se que todas elas indicam que uma base de dados para aplicação em engenharia necessita de um conteúdo semântico muito maior do que o admitido pelos modelos atuais, e daí este trabalho propor um novo modelo de dados,

apresentado no capítulo III, que permite representar o conteúdo semântico necessário.

II.3.1. - Sistemas de Bases de Dados para Aplicações não Convencionais.

Os Sistemas de Gerenciamento de Base de Dados atuais foram desenvolvidos, tendo em vista aplicações ditas comerciais, tais como Sistemas de Contabilidade, Folha de Pagamento, Registro de Transações, etc, e são na maior parte das vezes empregadas para esses fins. Esses empregos, que serão considerados convencionais, contrastam em termos de necessidades com os empregos visando outros objetivos, que serão considerados não convencionais, e que atualmente vêm encontrando mais e mais aplicações, o que vem forçando o estudo de novas soluções para os novos problemas que surgem.

Entre os empregos não convencionais de Sistemas de Gerenciamento de Bases de Dados, estão começando a ser estudados empregos em áreas como Bases de Dados para aplicação em Automação de Escritórios, para aplicações Gráficas, para aplicações Científicas e Estatísticas, para aplicações em Engenharia, e para a pesquisa em Inteligência Artificial. As características de cada uma dessas aplicações serão sumariamente expostas a seguir.

II.3.1.1. - Automação de Escritórios.

Automação de Escritórios é um termo que corresponde ao uso de sistemas computadorizados para auxiliar as atividades de um escritório, ou secretaria. Quatro áreas distintas podem ser identificadas em escritórios de uma organização [GAR_85]:

- Tarefas Administrativas, envolvendo atividades como Edição de textos, arquivamento, e recuperação de documentos, tratamento da correspondência, catalogação, etc;
- Comunicação, envolvendo atividades de troca de informação entre os departamentos de uma organização ou entre organizações, e envolvendo recursos como Mala eletrônica, transferência de arquivos e documentos, empregando variados meios, como telefone, telex ou serviços de fac-simile;
- Resolução de Problemas, que corresponde à criação de novos documentos, a montagem de relatórios a partir de informações de várias fontes, elaboração de mapas de andamento de atividades, etc;
- Absorção de conhecimento, que corresponde ao armazenamento de experiências e à criação de processos que garantam a continuidade das tarefas e experiências mesmo na ausência de pessoas chaves do escritório.

Para essas aplicações, as necessidades de uma base de dados para um escritório se concentram principalmente na armazenagem de documentos, os quais têm os mais variados formatos, desde textos datilografados e representação de fórmulas, até a de logotipos,

figuras, padrões de diagramação, etc. Além disso, é necessário que: além dos documentos em si, informações sobre esses documentos sejam mantidas e correlacionadas; seja mantido um registro de índices e catalogação desses documentos; sejam mantidas informações sobre a maneira como os elos de comunicação podem ser acionados; como a informação deve ser configurada para ser transmitida, em cada meio de comunicação disponível; sejam mantidas as várias versões de um documento, e qual o estado de desenvolvimento, segurança e cópias existentes de cada um.

O tratamento dado às necessidades de automação de escritórios atualmente se concentra principalmente nos problemas de transmissão e codificação para transmissão de documentos, e não nos sistemas de armazenagem desses documentos, para os quais ainda se procura utilizar os modelos de dados tradicionais, associados a esquemas de uso de sistemas de tratamento de arquivos, sem que se integrem os três problemas - codificação, armazenagem e tratamento - em um modelo único.

II.3.1.2. - Aplicações Gráficas.

Para Aplicações Gráficas, um Sistema de Gerenciamento de Base de Dados deve ser capaz de armazenar as informações gráficas, com outras informações não gráficas, mantendo a mesma estrutura e filosofia de acesso que um sistema necessita. Uma necessidade específica de um SGBD para aplicações gráficas, é a existência de várias formas de se representar a mesma informação gráfica, várias delas usadas simultaneamente por uma mesma apli-

cação, e a decorrente necessidade que um SGBD tem de reconhecer cada uma delas, e de identificar uma figura mesmo que ela se apresente em diferentes representações.

A especificação das necessidades de SGBDs para aplicações gráficas tem sido estudada, porém em geral visando usar modelos de dados já existentes, notadamente o Modelo Relacional, tal como em [CHA_80] e [BEC_80]. Devido ao fato que essas análises se concentram no emprego de um determinado modelo para aplicações gráficas, as necessidades geralmente estabelecidas somente se referem aos fatos que os modelos usados não conseguem representar a contento, deixando-se de mencionar aqueles que poderiam ser melhor representados num modelo que fosse a priori desenvolvido para essa aplicação.

As necessidades mais freqüentemente citadas, que os modelos de dados tradicionais não conseguem representar são:

- possibilidade de identificação de uma mesma informação gráfica representada em mais de uma maneira;
- identificação da ordem de ocorrência das várias figuras, e dos vários símbolos dentro de uma figura;
- a existência de mais de um ponteiro que possa indicar onde as próximas operações na base irão ocorrer, e poder fazer a indicação do local de uma operação como uma referência a qualquer dos ponteiros previamente estabelecidos;
- as operações de navegação da base de dados devem permitir o avanço e retrocesso de qualquer ponteiro, e com um deslocamento de um ou mais registros numa única operação;
- possibilidade de efetuar um particionamento dos dados

armazenados, permitindo a uma dada aplicação - que pode partilhar a base de dados com outras aplicações simultâneas - reservar uma partição da base para seu acesso exclusivo durante um período de tempo que pode ser elevado;

- possibilidade de armazenar o significado gráfico de um símbolo, de forma a permitir que o próprio SGBD efetue as operações comumente requeridas em um sistema gráfico;
- possibilidade de identificar cada registro (ou tupla) através de um único identificador mnemônico;
- existência de mais de um registro repetido (ou tupla) em um mesmo arquivo (ou relação).

II.3.1.3. - Aplicações Científicas e Estatísticas.

As Bases de Dados para Aplicações Científicas e Estatísticas (BDACE) tem como atividade principal a armazenagem dos dados obtidos por experimentos científicos e por amostragens de dados. As aplicações típicas dessas bases de dados correspondem por exemplo à armazenagem de dados meteorológicos, telemetria, recenseamentos, coleta de dados de experimentos científicos, etc.

As aplicações científicas e as aplicações estatísticas são similares no que diz respeito às necessidades de suporte por um SGBD [SHO_85], daí o fato de serem estudadas conjuntamente. Além disso, em geral são necessárias operações estatísticas sobre os dados obtidos de um experimento científico, se bem que em

geral as aplicações científicas requerem um tratamento mais cuidadoso, e são em geral mais complexas do que as que tratam estritamente de dados estatísticos.

As BDACE possuem algumas características que são marcantes para tais aplicações. Uma delas é o grande volume de informações manipulado. Por exemplo, os dados necessários para uma simulação meteorológica pode envolver milhões de dados; as coletas de dados sobre o tempo são obtidas sobre toda a superfície terrestre, e são atualizadas várias vezes por dia. O tratamento das informações sobre doenças, armazenadas em centenas de hospitais são analisadas e comparadas para se traçar estatísticas e para se inferir dados para diagnósticos somente possíveis através da correlação de milhares de casos e com centenas de variáveis.

Os dados obtidos pelos experimentos científicos são também freqüentemente esparsos. As bases de dados empregadas em aplicações comerciais podem ser vistas como tabelas, em que a grande maioria, quando não todas as colunas, contém dados significativos. As BDACE ao contrário, são freqüentemente constituídas de matrizes muito grandes mas também muito esparsas, ou seja, matrizes em que a maior parte de suas posições não contém valores conhecidos ou definidos.

Outra característica própria das BDACE levantado em [SHO_85] é o fato de que os dados científicos freqüentemente apresentam uma característica de "Multidimensionalidade". Esse termo se refere ao fato que muitos dados dependem diretamente de outras informações que não os próprios dados. Por exemplo, um experimento científico pode ser repetido muitas vezes, variando-se as condições em que esses experimentos são repetidos, tais como

temperatura, pressão, ou qualquer outro parâmetro condizente com o caso particular. Para cada conjunto de parâmetros os dados medidos no experimento são repetidos. Diferentemente dos sistemas convencionais de bases de dados, em que os conceitos de registro ou entidade são suficientes para a identificação e as operações sobre os dados, em aplicações científicas é mais conveniente que tratem-se os resultados de cada experiência como "casos", em que a identificação dos casos é feita pelos parâmetros usados para se obter as medidas identificadas.

Considerando-se o emprego do Modelo Entidade-Relacionamento, pode-se às vezes fazer a identificação desses casos como atributos de um relacionamento, sendo que o relacionamento envolve as entidades que correspondem aos parâmetros usados. No entanto surgem daí alguns problemas. O primeiro deles é que passa a não haver uma identificação imediata na base de dados para cada caso de medida, ao passo que o usuário normalmente pode se referir àquelas medidas através da indicação dos parâmetros usados para obtê-las. O segundo é que nem sempre os parâmetros que são mudados de medida para medida mantêm-se estáveis, isto é, além de algumas combinações de valores não fazerem sentido, ocorre que freqüentemente intervêm parâmetros que em outras medidas têm significado diferente, ou nem mesmo são parâmetros. Por exemplo, em um recenseamento, certas informações somente são aplicáveis para uma classe de pessoas, não tendo nenhum significado para outras. Freqüentemente a própria escolha do que é uma entidade deixa de ter significado, quando hierarquicamente uma entidade nada mais é do que um tipo de outra entidade.

Dessa discussão, podem ser identificadas como principais necessidades de uma Base de Dados para Aplicações Científicas e

Estatísticas:

- manipulação de volume de dados muito grande, frequentemente distribuídos entre diversos sistemas de computação;
- possibilidade de compressão de dados esparsos, e acesso e manipulação eficiente sobre esses dados comprimidos. É desejável que se possa realizar operações mais frequentemente usadas sobre os dados comprimidos, e não que seja necessário antes descomprimi-los para então serem operados;
- existência de estruturas de dados que sejam capazes de representar a multidimensionalidade de identificação de medidas discutida;
- possibilidade de armazenar Meta-dados em grande quantidade e em variadas formas não estruturadas sobre as informações armazenadas, usando modelos de muito conteúdo semântico;
- possibilidade de operações relacionais sobre as informações, tais como as tradicionais operações de seleção com partição, união e intersecção, porém levando em conta os meta-dados armazenados exteriormente às relações existentes;
- existência de operações especiais sobre os dados, tais como interpolação, mudança de unidades, agregação de dados (somatório, média, operações estatísticas), projeção de dimensões e navegação em informações esparsas.
- Armazenagem de tipos de dados complexos, como matrizes, informações gráficas, informações temporais, etc.

- uso de "Hardware" especial, que pode ser necessário para a realização dos experimentos, mas que pode ser aproveitado também para o Sistema de Gerenciamento da Base de Dados.

II.3.1.4. - Pesquisa em Inteligência Artificial.

Nas discussões anteriores deste capítulo, os termos "dado" e "informação" têm sido usados segundo conceitos que originam-se das pesquisas atualmente em curso na área de Inteligência Artificial. Neste ponto convém que esses conceitos sejam explicitados, juntamente com o conceito de regra.

O termo "Dado" é usado quando se quer referir à um valor, no mesmo sentido que no Modelo Entidade-Relacionamento usa-se o termo "Valor de Atributo". Deve-se ressaltar que valor pode não ser apenas um número, mas uma cadeia de caracteres ou uma matriz, mas sempre determinado.

O termo "Informação" é usado quando se quer referir ao significado do valor, no mesmo sentido que no ME-R faz-se referência aos dados conhecidos sobre "Atributos", os quais correspondem aos meta-dados dos dados armazenados.

O termo "Regra" refere-se à especificações conhecidas sobre as informações disponíveis. Pode tanto significar as restrições que impõem a consistência em uma base de dados tradicional, quanto as condições que permitem que se estabeleçam inferências sobre os dados em uma base de conhecimento [DEB_85].

Por exemplo, a frase "o número de páginas de uma referência não pode ser inferior a 5 e nem superior a 100" constitui uma

regra. A frase "o número de páginas da referência Título Projeto-de-Base-de-Dados é 36" constitui uma informação. E "100", "36", "5" e "Projeto-de-Bases-de-Dados" são dados.

Na terminologia empregada em inteligência artificial, costuma-se chamar "Base de Conhecimento" ao conjunto de regras e informações que uma aplicação utiliza, bem como "Sistema Perito" ao programa de aplicação. O termo "Sistema de Gerenciamento de Conhecimento" [KLG_86] tem sido proposto para designar a integração de um Sistema de Gerenciamento de Bases de Dados com um correspondente Sistema de Gerenciamento de Regras, que forneceria todos os dados, informações e regras que o Sistema Perito necessitasse. Têm sido realizados estudos, e argõe-se que é necessário o surgimento de Sistemas de Gerenciamento de Regras com o mesmo grau de flexibilidade e sofisticação atualmente apresentado pelos Sistemas de Gerenciamento de Bases de Dados.

Considerando-se as necessidades de um sistema de Gerenciamento de Conhecimento, integrando Dados e Regras, pode-se estabelecer que é necessário principalmente que:

- exista uma representação abrangente e expressiva de conhecimento. Essa representação deve ser flexível o suficiente para permitir um "rearranjo" do conhecimento, isto é a mudança de um sub-modelo do conhecimento armazenado, sempre que um fator externo requisitar essa mudança. Por exemplo, sempre que um novo fato, anteriormente não considerado, for introduzido, todo o conhecimento que pode ser afetado por esse novo fato deve poder ser "repensado";
- a estrutura "bem comportada" e repetitiva dos dados, reconhecida pelos SGBDs possa ser integrada às formas

de representação de conhecimento adotadas. Um uso que tem sido generalizado é o emprego do Modelo Relacional para armazenar relações de dados encaradas como instâncias de predicados do cálculo relacional. Por exemplo, considera-se que o predicado referência-autor(x,y), que simboliza ser y autor da referência x , pode ter suas instâncias armazenadas em uma Base de Dados Relacional em que existe a relação Autores(Referência, Autor), sobre a qual os quantificadores do cálculo de predicados podem ser verificados. No entanto, dessa forma a representação do predicado em si é totalmente diferente da representação de suas instâncias, e uma maior integração deve ser buscada;

- deve existir uma forma bem determinada de distinguir "informação" de "conhecimento" (regras), sem contudo tornar diferentes as formas de análise de informações e de conhecimento;
- deve ser possível ao próprio sistema desenvolver o conhecimento obtido, ou seja deve ser possível ao sistema detectar que informações podem ser substituídas por regras, e quando regras podem produzir novas informações;
- a verificação de regras frequentemente é feita de uma maneira explosiva, ou seja, o resultado da verificação de uma regra frequentemente pode levar à necessidade de verificação de várias outras regras. Por isso, o Sistema de Gerenciamento de Conhecimento deve poder aproveitar qualquer possibilidade de processamento paralelo que o sistema de computação hospedeiro puder

oferecer. Especialmente, é desejável que a exploração de regras e desenvolvimento automático de conhecimento possa ser efetuado sem prejudicar a execução do programa de aplicação (o sistema perito) que está usando o Sistema de Gerenciamento de Conhecimento.

11.3.2. - A Caracterização de uma Base de Dados de Engenharia.

Com a proliferação de sistemas distribuídos, muitas aplicações freqüentemente operam em máquinas dedicadas, mas com a troca de informações através da rede onde estão integradas. É comum a existência de redes, nas quais as informações armazenadas correspondem às mais diversas aplicações. No entanto, apesar da necessidade de comunicação entre as várias aplicações, as informações usadas em cada aplicação freqüentemente estão armazenadas usando modelos diferentes em cada aplicação, fazendo com que a comunicação somente possa ser feita através de interfaces especialmente desenvolvidas para cada caso em particular.

Tal estado de coisas faz com que surja a necessidade de integração de todos os dados de qualquer tipo de aplicação em uma estrutura unificada (ou SGBD), que atenda aos requisitos básicos de todas as aplicações envolvidas.

Assim, além da pesquisa para se obter SGBDs que supram as necessidades de cada tipo de aplicação não convencional, procura-se também fazer com que as soluções encontradas possam ser aplicadas em áreas correlatas, aumentando o espectro de abrangência do sistema, e permitindo a integração dos dados e informações de outras aplicações no mesmo modelo.

Devido ao amplo espectro de necessidades que uma aplicação de engenharia necessita, vem se tornando popular a designação de Sistema de Gerenciamento de Dados para Engenharia, dada a sistemas cuja capacidade de representação envolve necessidades de várias aplicações não convencionais, além das convencionais.

Um Sistema de Gerenciamento de Dados para Engenharia (SGDE) pode ser visto como um sistema que apoie o processo de projeto de algum novo produto, seja ele um programa de computador, um circuito integrado ou uma peça mecânica. Nessa linha de raciocínio, pode-se dizer que um SGDE é um sistema de Base de Dados para dar apoio a sistemas de Projeto Auxiliado por Computador (PAC), ao qual se integram sistemas de Fabricação Auxiliada por Computador (FAC) [WSU_86], conhecida do inglês como CAD/CAM. Ou seja, pode-se dizer que um SGDE é um SGBD para PAC/FAC integrado.

Uma Base de Dados para uma ferramenta de auxílio ao projeto é muito mais um objetivo a ser construído, do que um catálogo estático de dados a que a ferramenta deve consultar, tal como é o caso das aplicações tradicionais de Bases de Dados. Essa característica "Dinâmica" que pode ser associada às Bases de Dados de Engenharia está associada não às atualizações dos dados da base de dados em si, que mesmo numa aplicação tradicional pode ocorrer a cada milissegundo, mas à estrutura da Base de Dados, que raramente muda em uma aplicação convencional, mas muda constantemente em uma aplicação em que o desenvolvimento de algo novo pode corresponder ao próprio desenvolvimento da estrutura da Base de Dados.

Dessa forma, para uma aplicação em Engenharia, o próprio esquema conceitual da Base de Dados usada pode ser alterado como parte do processo de desenvolvimento, sendo necessária a

existência de um Meta-esquema conceitual, que estabeleça como o esquema pode ser alterado sem que seja necessária uma reformulação da estrutura de armazenagem da própria Base de Dados já inicializada.

Considera-se assim que o projeto resultante é a própria Base de Dados [MAT_81], a qual pode ser submetida ao sistema de FAC integrado.

Um projeto de engenharia pode muitas vezes ser considerado como o projeto de algo que tem uma forma geométrica definida. Por exemplo, no projeto de um circuito integrado, as várias formas que o projeto pode ser visto (geometria do lay-out, esquema em nível de transistores, esquema lógico, etc.), todas elas têm uma representação geométrica. Dessa forma, um Sistema de Gerenciamento de Dados para Engenharia incorpora todas as características de um SGBD para aplicações gráficas.

O projeto de um produto de engenharia é um processo de refinamento sucessivo, em que os conceitos imaginados são transformados em objetos reais, as especificações genéricas são detalhadas, o todo é analisado em partes até os menores detalhes. No processo de projeto, o engenheiro parte de especificações genéricas ou de produtos realizados anteriormente, e vai sucessivamente incorporando novos dados, ou modificando partes já definidas anteriormente. Uma ferramenta de apoio ao projeto deve verificar a consistência das novas informações e das mudanças efetuadas, comparar com as restrições impostas ao projeto, usando todas as informações presentes na Base de Dados que sejam correlatas a cada passo do processo.

Muitas versões de um projeto podem existir simultaneamente, e essa característica traz a necessidade de um SGDE poder rea-

lizar o gerenciamento de múltiplas versões e alternativas propostas para um projeto. Katz em [KAT_84] caracteriza "Versão" e "Alternativa" de um projeto, considerando que um projeto está inicialmente em progresso, e sobre esse projeto os projetistas efetuam mudanças e incorporação de dados. As alterações feitas para experimentar outras formas de implementação são Alternativas, e as alterações que são feitas sobre um projeto já aprovado anteriormente são chamadas versões. Na seção II.6.3 esse assunto é tratado em maiores detalhes. Note-se que um projeto pode ter muitas versões e muitas alternativas, sendo que as versões produzidas são armazenadas por tempo indefinido.

Um Sistema de Gerenciamento de Dados para aplicações de engenharia deve portanto dar suporte à armazenagem das várias versões e alternativas de um projeto. A forma imediata de se apoiar essa necessidade é permitir a cópia de informações, mantendo um conjunto de dados em que cada elemento constitui uma versão ou alternativa de projeto. Esse enfoque no entanto, não é adequado, pois torna cada versão separada das demais, quando na realidade elas estão intimamente ligadas, perdendo-se a informação que une cada versão.

O problema de controle das versões produzidas por um processo de desenvolvimento de software tem sido estudado sob a denominação de Gerenciamento de Configuração de Software [BER_84], em que, a menos do fato de não envolver o conceito de "Alternativa", trata do mesmo problema, apenas restrito ao desenvolvimento de software. Os conceitos de Gerenciamento de Configuração de Software, discutidos na seção II.6.3 originaram-se do problema de gerenciamento de configurações em geral, as quais foram re-estudadas sob o enfoque das necessidades de configurações (ver-

sões) de projeto de software. No entanto, os conceitos desenvolvidos podem ser aplicados ao Gerenciamento de Configurações de Projetos em geral, e serão aplicados para o desenvolvimento deste trabalho como um recurso de auxílio ao gerenciamento e controle de desenvolvimento de projetos em geral.

O reconhecimento da existência de uma fase de teste de um projeto proposto leva à necessidade de se estabelecer como uma ferramenta de apoio ao projeto poderia auxiliar também a realização de testes.

O teste de um particular objeto de projeto é feito submetendo-se esse objeto ao crivo de um conjunto de regras que irão verificar a integridade, completeza e consistência do objeto com as especificações de projeto. Como o objeto, ou uma sua representação em uma Base de Dados é necessariamente diferente da representação das especificações, geralmente se admite que o procedimento de teste deve envolver pessoas ou então a atuação de um sistema perito, que contenha conhecimento sobre o processo de teste.

Sob esse enfoque, um Sistema de Gerenciamento de Dados de Projeto integrado deve ser capaz de armazenar, além do projeto e de suas especificações, também as regras necessárias à atuação de um sistema perito. Dessa forma, um SGDE deveria idealmente incorporar todas as características necessárias a um SGBD para uso em Inteligência Artificial.

Freqüentemente um trabalho de engenharia pode necessitar de especificações obtidas através de coleta de dados. Em particular, um sistema pode ser usado para obter dados de testes, seja o teste aplicado sobre o próprio produto sendo desenvolvido, ou sobre alguma entidade que propicie obter dados para o desenvol-

vimento. As necessidades que se apresentam para um Sistema de Gerenciamento de Base de Dados que dê suporte a isso são as mesmas de um que dê suporte a aplicações científicas e estatísticas, e portanto de uma maneira geral, um sistema de apoio à projetos poderá necessitar manipular informações tipicamente mantidas por Bases de Dados para aplicações Científicas e Estatísticas.

Uma Base de Dados para Engenharia também irá necessitar armazenar documentos de referência de experiências anteriores, manuais e normas de projeto, ou seja, de uma forma geral, documentos nos mais variados formatos, que serão tanto usados pelos projetistas quanto poderão ser criados como consequência do processo de projeto. Dessa forma, mais uma vez, um SGDE deveria poder incorporar as características de um SGBD para aplicações em Automação de Escritório, que é tipicamente o termo empregado para as tarefas de Catalogação, Armazenagem e Preparo de documentos.

Pode-se notar então, que um sistema integrado de projeto e produção, como atualmente tem se mostrado necessário, para poder dispor de uma estrutura homogênea de representação de toda a informação armazenada e manipulada por um tal sistema, irá requerer do Sistema de Gerenciamento de Dados que o suporta, capacidade de suprir todas as necessidades de sistemas para aplicação em automação de escritórios, aplicações gráficas, aplicações científicas e estatísticas e aplicações de inteligência artificial.

A complexidade de um tal sistema é muito grande, e atualmente não existe nenhum sistema que possa dar suporte a toda essa gama de necessidades. No entanto, tendo sido reconhecidas, pode-se procurar avançar na direção de se encontrar um sistema

que seja ao menos mais adequado que os sistemas atuais, para suprir todas essas necessidades.

As observações acima permitem identificar ao menos um conjunto mínimo de necessidades que um Sistema de Gerenciamento de Dados para Engenharia deverá suprir:

- o sistema de dados deverá atender aplicações com necessidades bastante diversas. Portanto deverá poder fornecer e receber os dados para e de cada aplicação em formato adequado a cada aplicação; (n1)
- muitos projetistas irão participar de um mesmo projeto, e freqüentemente simultaneamente através de várias estações de trabalho que podem ter capacidade de processamento local. A Base de Dados deve então ser distribuída e admitir pesado acesso concorrente. Além disso, como cada estação de trabalho trata de uma parte geralmente bem definida dos dados armazenados, essa partição naturalmente estabelecida sobre os dados deve ser reconhecida pelo sistema; (n2)
- o esquema conceitual da Base de Dados muda dinamicamente. Portanto deve ser possível a especificação de um Meta-esquema de dados, que oriente as alterações do esquema de dados; (n3)
- além dos dados, grande quantidade de informações sobre os dados (Meta-dados), informações essas altamente não estruturadas, deverão ser armazenadas. O conteúdo semântico capturado pelo modelo utilizado deverá ser então muito alto; (n4)
- a existência de versões diferentes e simultâneas do projeto ou de partes do projeto indica a necessidade da (n5)

existência de um gerenciamento de versões, em que grandes quantidades de dados repetidos se alternam com inter-relacionamentos únicos entre elementos desses dados;

- a identificação das várias versões e dos dados obtidos por coleta de dados devem poder ser feitas tanto por referência direta quanto através dos relacionamentos que cada agrupamento de dados mantém com outros dados da base; (n6)
- mais de um projeto pode ser desenvolvido simultaneamente, sendo um independente de outro, mas freqüentemente necessitando de dados em comum. Assim, Bases de Dados para diferentes projetos devem coexistir e propiciar inter-relacionamentos consistentes; (n7)
- muitos tipos de dados são usados pelas diferentes aplicações que se utilizam da Base de Dados. Devido a isso, uma ampla gama de tipos de dados deve poder ser atendida; (n8)
- em especial, todas as necessidades de uma base de dados para aplicações gráficas são necessárias em uma base de dados para aplicações em engenharia. (n9)
- engenheiros, cientistas e pessoas envolvidas em projetos de sistemas em geral não são necessariamente pessoas experimentadas, nem necessitam de conhecimentos profundos sobre SGBDs para necessitarem deles. Dessa forma, as Bases de Dados para aplicações nessa área devem poder ser amigáveis, e acessíveis mesmo para os mais inexperientes usuários. (n10)

Essas necessidades serão novamente abordadas no capítulo III, deste trabalho, onde se apresentará a proposta de um novo modelo de dados que poderá ser usado para o desenvolvimento de um Sistema de Gerenciamento de Dados de Engenharia que atende as necessidades apresentadas. No capítulo IV, os vários tópicos levantados serão novamente estudados, já então com o objetivo de analisar como o modelo proposto pode atendê-los.

II.4. - Modelos não Convencionais de Interesse para Aplicações em Engenharia

A necessidade de se poder representar mais detalhadamente diversas situações do mundo real, tem motivado o surgimento de vários Modelos de Dados para Bases de Dados, todos eles procurando de alguma forma estender o refinamento da representação dos dados que é possível representar através de cada novo modelo. A tendência geral é evitar que apenas tabelas de dados (registros, relações) possam ser armazenadas, mas também dados menos "bem comportados".

A conscientização é geral quanto a que o aumento da complexidade das estruturas de armazenagem leva a uma menor eficiência das operações de manipulação dos dados da base de dados, porém reconhece-se também que com a representação mais detalhada do mundo real, muitas aplicações podem passar a concentrar-se apenas nos dados que efetivamente necessita, facilitando o seu desenvolvimento e operação, compensando e até excedendo a diminuição de eficiência do SGBD.

A seguir serão apresentados os modelos de dados mais signi-

ficativos, que têm sido propostos e experimentados com esse objetivo.

II.4.1. - Abstrações de Generalização e Agregação.

Smith e Smith em [SMI_77a] e [SMI_77b] desenvolvem o conceito de abstração de dados em uma base de dados através de duas formas de abstrações: Agregação e Generalização. Essas duas formas são apresentadas como extensões ao modelo relacional de Codd, e não como um novo modelo de base de dados. No entanto, o conceito de abstração de dados aumenta a capacidade semântica do modelo, e pode-se considerar que um modelo relacional acrescido de capacidade de representação de abstrações de Agregação e Generalização passa a ser um modelo semântico com capacidade de representação semântica consideravelmente maior. Esses dois trabalhos têm sido a base de desenvolvimento de todos os modelos propostos que têm por base o modelo relacional.

Os autores propõem o reconhecimento de abstrações em um modelo como a omissão deliberada de certos detalhes, com o objetivo de conseguir-se um modelo que represente os fatos do mundo real de uma maneira "intelectualmente gerenciável", e propõem que isso seja feito através de uma hierarquia de abstrações.

Uma abstração de agregação é descrita como um relacionamento entre um conjunto de itens ("Individuals") que é visto como um outro item de mais alto nível. Considerando-se o item de nível mais alto, muitos detalhes do relacionamento podem ser temporariamente ignorados.

Uma abstração de generalização ocorre quando um conjunto de itens similares são vistos como um único item genérico. Considerando-se o item genérico, as diferenças individuais de cada objeto específico são temporariamente ignoradas.

Os autores chamam a atenção para o fato de que os conceitos de abstração e de normalização de relações são dois conceitos complementares, ressaltando o fato que o desmembramento de uma relação em outras para efeito de se normalizar um esquema não constitui uma agregação, embora a identificação de agregações possa levar a uma normalização. Para enfatizar a diferença entre os dois conceitos, ressalta-se o fato de que uma abstração tem um sentido lógico, e portanto pode e deve ser identificada por um nome significativo para o Administrador da Base de Dados.

Sobre o conceito dessas duas formas de abstração são desenvolvidas cinco regras semânticas que são consideradas invariantes em todo modelo relacional que usar pelo menos uma dessas formas de abstração. É construído também um modelo sintático de uma linguagem que pode especificar as construções do modelo envolvendo agregações e/ou generalizações.

Através da formalização das regras que definem a sintaxe para a descrição de abstrações, torna-se claro que a existência de agregações no modelo relacional é possível desde que o Sistema de Gerenciamento de Dados mantenha a responsabilidade de manter duas relações lógicas para cada abstração que for especificada, as quais mantêm em seus respectivos níveis de abstração os atributos próprios de cada nível, repetindo (e é nesse ponto que o SGBD deve garantir a consistência dos dados nas duas relações) os atributos que são comuns aos dois níveis, dos quais os atributos de identificação são necessariamente repetidos.

Além disso, percebe-se que é possível a especificação de uma abstração de agregação independente de uma generalização, porém uma abstração de generalização sempre implica em uma agregação. Isso decorre do fato de que a indicação de a qual relação de itens específicos um dado item genérico se refere, é feita através de um atributo associado ao item específico, que constitui assim uma agregação das informações específicas àquele item.

E' proposta uma forma gráfica de representação das duas formas de abstrações, que estabelece dois planos perpendiculares para a representação dessas duas formas. Considera-se que no plano do papel são representadas as abstrações de agregação, usando para isso uma notação derivada da Notação de Bachman, onde a hierarquia de agregações aparece com o objeto de maior grau de abstração no topo da página. Para cada relação que represente uma abstração de generalização estariam representados em um plano perpendicular à página as relações específicas da qual a relação é uma generalização. Note-se que os valores do domínio do atributo que é usado para efetuar-se a generalização são discretos, e portanto a representação gráfica envolvendo um plano perpendicular ao papel é na realidade constituída de vários planos paralelos ao papel justapostos, um para cada valor de atributo possível.

Os conceitos de abstração propostos nos trabalhos citados permitem a representação de uma grande quantidade de informações semânticas, e por isso, apesar de não caracterizar um novo Modelo de Dados para Bases de Dados, os conceitos de abstração de agregação e abstração de generalização de Smith & Smith são uma ferramenta de modelagem semântica bastante importante.

II.4.2. - O Modelo Relacional Extendido - RM/T.

Tendo sugerido em 1970 o Modelo Relacional, Codd apresentou em [COD_79] um conjunto de sugestões para aumentar a capacidade de representação semântica de um esquema de dados que tivesse como objetivo ser implantado em um Sistema de Gerenciamento de Bases de Dados usando o Modelo Relacional. Essas sugestões passaram a ser conhecidas como Modelo Relacional Extendido ou Modelo Relacional/Tasmânia (local de um congresso onde as idéias foram originalmente apresentadas).

A essência desse modelo consiste no uso de um SGBD relacional convencional, ao qual é acrescido um Dicionário de Dados que cuidaria de algumas relações especiais armazenadas na Base de Dados com o propósito de fornecer informações sobre o significado das demais relações. Os dados seriam então "Filtrados" pelo Dicionário, usando as relações especiais, antes que alguma alteração na Base de Dados fosse autorizada.

Esse enfoque apresenta algumas vantagens, entre elas a possibilidade de armazenar de uma forma padrão, na mesma base de dados, tanto os dados alvo de cada aplicação, como os Meta-dados (informações semânticas) sobre esses dados, permitindo por exemplo que uma aplicação indicasse por si própria alterações no esquema de dados, usando uma Linguagem de Descrição de Dados idêntica à Linguagem de Manipulação de Dados. No entanto apresenta como desvantagem o fato que não existe na estrutura de dados nenhum elemento intrínseco que garanta algum tipo de integridade estrutural na descrição, dificultando a manutenção da integridade dos dados armazenados.

O autor apresenta um conjunto de conceitos destinados a aumentar a capacidade de representação do modelo. Fundamental entre todos eles, está a constatação de que é necessário que o SGBD reconheça em todas as relações, quais são as entidades do mundo real que cada registro de uma ou mais relações estão efetivamente modelando, considerando que os próprios atributos identificadores de cada relação são inadequados para isso, uma vez que admite-se que o usuário pode alterar a identificação de uma entidade, sem que a entidade em si possa ser alterada. Cita como exemplo o nome ou o número de registro de um navio, que em situações especiais podem ser alterados, sem que o navio se transforme em outro. Para evitar isso, o autor sugere que para todos os tipos de entidades do mundo real efetivamente modeladas, seja mantido uma relação especial que forneça para cada entidade modelada uma identificação controlada pelo SGBD que o usuário não pode alterar, a menos que indique explicitamente alteração na entidade do mundo real. Com esse conceito adicional, o RM/T passaria a ter em todas as relações que se refiram a uma dada entidade, a indicação do identificador interno que corresponde a cada entidade representada. Esse novo atributo é chamado pelo autor como o "designador" ("surrogate") de cada entidade, e cada relação criada para manter os designadores de cada entidade são chamadas de E-Relações.

As relações que efetivamente armazenam dados, aquelas existentes também em um sistema relacional não estendido, são chamadas pelo autor de P-Relações (Relações de Propriedades).

As entidades de um sistema descrito com o RM/T podem ter um subtipo além de um tipo, desde que mantenha-se a regra de que se uma entidade é de um subtipo de um determinado tipo, então ela é

também daquele tipo, e todos os atributos que são válidos para as entidades de um determinado tipo são também válidos para todas as entidades de seus subtipos. O autor classifica todas as entidades e os seus tipos representados em:

- Entidades Características - aquelas que são subordinadas à existência de algum outro tipo;
- Entidades Associativas - aquelas que representam relacionamentos entre entidades de outros tipos;
- Entidades do Núcleo - aquelas que existem por si só, independentemente das demais.

Tendo sido feita a introdução desses conceitos, que são mantidos em cada Base de Dados através de E-Relações, o autor acrescenta várias outras relações especiais, cada uma destinada a armazenar informações sobre uma possível forma de interação entre cada um dos tipos de dados que uma determinada Base de Dados contém. Assim existem relações de todos os tipos de entidades, relações que associam cada subtipo ao seu tipo imediatamente superior (uma vez que podem existir hierarquias de tipos e subtipos), relações que indicam quais são as P-Relações de cada tipo, relações de todos os relacionamentos entre entidades, etc.

II.4.3. - O Modelo Funcional.

O Modelo Funcional tem por objetivo a representação de dados usando um enfoque funcional como ferramental para a modelagem de dados de uma aplicação. Algumas implementações usando o modelo funcional têm sido efetuadas, ainda em fase de protótipos.

O Modelo Funcional tem como básicos os conceitos de Entidade e de Função. Entidades são todos os objetos, abstratos ou reais, do mundo real sobre os quais exista interesse de que sejam representados em uma base de dados. As Funções se aplicam sobre as entidades, e têm como resultado também Entidades. As Entidades são Criadas em um modelo por uma ação de criação explícita, que indica o início de interesse no objeto do mundo real correspondente, e existe até que uma ação explícita solicite a retirada dessa entidade do modelo, que significa o fim desse interesse. Tipos de dados básicos tais como Entidades, Números-Inteiros e Cadeias-de-Caracteres são também considerados entidades, porém mantêm a propriedade única no modelo de existirem pré-definidos, e de não poderem ser retirados.

Uma Função pode ter qualquer número de argumentos. Uma Função que não tenha nenhum argumento indica um tipo ou a ocorrência de um subtipo de uma entidade. Por exemplo, as declarações:

PESSOA() -> Entidade

ALUNO() -> PESSOA

especificam que PESSOA é uma entidade, e que ALUNO é um subtipo de PESSOA. Os tipos e subtipos formam uma hierarquia, onde as funções que se aplicam a um tipo também aplicam-se a todos os seus subtipos, o que permite um mecanismo de "herança" de propriedades.

Funções que tenham um argumento indicam propriedades da entidade que é argumento da função. Por exemplo, as declarações:

NOME(PESSOA) -> Cadeia-de-Caracteres

PROFESSOR(ALUNO) -> PESSOA

indicam que o NOME de uma PESSOA é uma cadeia de caracteres, e que o PROFESSOR de um ALUNO é uma PESSOA.

Funções que tenham mais de um argumento podem ser usadas para expressar as propriedades que se associam a interações entre entidades (semelhante aos atributos de relacionamentos no ME-R). Por exemplo, com as declarações:

CURSO() -> Entidade

NOTA(ALUNO, CURSO) -> Inteiro

fica estabelecido que a NOTA de um ALUNO em um CURSO é um valor inteiro.

O resultado de uma função pode ser apenas uma entidade, ou um conjunto de entidades, e nesse caso a propriedade que ela representa pode assumir múltiplos valores. Funções podem ter os seus valores declarados explicitamente, tal como as apresentadas acima, e são então chamadas de funções básicas. Outras podem ter os seus valores computados a partir de suas entidades argumento, quando então são chamadas de funções derivadas.

O Modelo Funcional tem por objetivo a modelagem de situações do mundo real, definindo uma linguagem para a especificação dos esquemas de dados, e sugerindo que linguagens de programação já existentes sejam estendidas com primitivas que permitam o acesso a uma base de dados que atenda as características do modelo.

Shipman em [SHI_81] propõe uma Linguagem de Definição e Consulta de Dados denominada DAPLEX, e apresenta um analisador capaz de reconhecer comandos nessa linguagem. O autor sugere que um SGBD relacional já existente poderia ser usado acoplado a esse analisador, para o uso do modelo funcional usando a linguagem DAPLEX. Para isso, sugere-se que as entidades e as funções bási-

cas de mais de um argumento sejam mapeadas em relações no SGBD utilizado, tendo como atributos as funções básicas que são definidas sobre as entidades e tuplas de argumentos, sendo que as entidades argumento seriam os atributos chaves de identificação das relações.

As informações que o Modelo Funcional apresenta como derivadas seriam então tratadas por um Dicionário de Dados que executaria as operações definidas em cada uma das funções derivadas, usando como valores os atributos das relações efetivamente armazenadas nas bases de dados.

Em [KUL_86] apresenta-se uma outra implementação do modelo funcional, ressaltando-se que a Linguagem de Definição e Consulta de Dados utilizada é a mesma linguagem DAPLEX definida em [SHI_81], porém expandida para englobar as estruturas e operações usuais na linguagem de programação ALGOL, criando uma nova linguagem denominada EFDM. O analisador dessa linguagem, do qual foi implementado um protótipo, acessa diretamente as estruturas de dados, executando então as funções de um interpretador de linguagem de programação e de um gerenciador de base de dados, sendo que esse gerenciador foi concebido desde o início para o modelo funcional.

II.4.4. - Modelos Semânticos.

Em determinadas aplicações em que é necessário usar bases de dados, a maneira homogênea com que as estruturas são tratadas pelos Sistemas de Gerenciamento de Base de Dados convencionais,

torna pequena a capacidade de representação de dados não estruturados, ou seja, aqueles que ocorrem esporadicamente, ou então aqueles que refinam casos particulares das ocorrências tratadas homogeneamente - que frequentemente são tratadas como Meta-Dados - ou Informações Sobre os Dados armazenados.

Para permitir estruturas de armazenagem de dados que permitam representar dados naturalmente mais heterogêneos do que aqueles que podem ser estruturados em tabelas, e aumentar a capacidade de se modelar Meta-dados, vários modelos têm sido propostos.

A seguir será feita uma caracterização do que se entende ser um modelo semântico, e é feita a distinção entre modelos semânticos e redes semânticas, dois termos parecidos para soluções diferentes de problemas diferentes que caminham na mesma direção.

A seguir serão apresentados modelos de bases de dados semânticas que nos últimos anos têm sido desenvolvidas para permitir a representação de maiores quantidades de informações semânticas do que a permitida nos modelos tradicionais. Deve-se notar, que entre os modelos apresentados, apenas um (SEDACO) tem efetivamente um protótipo operacional, sendo que os outros três apresentam-se como formalismos metodológicos para a preparação de esquemas conceituais, que devem depois ser transformados para os modelos dos Sistemas de Gerenciamento de Bases de Dados que serão efetivamente usados.

II.4.4.1. - Modelos Semânticos e Redes Semânticas.

Redes semânticas são estruturas amplamente usadas nas áreas de inteligência artificial que necessitam representar o conhecimento dos objetos manipulados através da indicação do que cada objeto é, em termos dos demais objetos presentes na rede. Uma grande parte do conhecimento que é armazenado em bases de conhecimento vem de sentenças tais como "Jose é carpinteiro", isto é, pode-se representar conhecimento explorando o fato de poder-se fazer a predicação de um indivíduo (tal como José), atribuindo-lhe um determinado tipo (carpinteiro). Outra grande quantidade de conhecimento advém do fato que se pode expressar o tipo de uma coisa (por exemplo carpinteiro) como sendo um subtipo de outro (por exemplo artífice). Essas duas formas de se expressar conhecimento podem ser modeladas estabelecendo-se ligações "E'" ("IS-A") entre os elementos que compõem cada sentença. Redes onde os nós são interconectados por ligações "E'" são denominadas Redes semânticas. Brachman apresenta em [BRH_83] um amplo conjunto dos significados que podem ser atribuídos a conhecimento expresso através de ligações "E'".

As conexões formadas por ligações "E'" caracterizam redes hierárquicas não estritas, isto é, apesar de um certo subtipo não poder ser subtipo dele mesmo (nem mesmo indiretamente), um subtipo pode ser subtipo de vários tipos diferentes. Essa hierarquia é feita através do refinamento sucessivo de tipos genéricos em subtipos cada vez menos genéricos, que podem "herdar" as propriedades dos tipos mais genéricos, sendo que no nível mais detalhado da hierarquia estão as ocorrências individuais dos objetos representados.

A principal diferença que existe entre as redes semânticas, usadas para armazenar conhecimento, e os modelos semânticos, usados para representar esquemas de dados em bases de dados, é justamente o fato que os modelos semânticos estabelecem o esquema de dados, sendo estes armazenados em uma estrutura diferenciada da usada para representar os próprios dados, enquanto em uma rede semântica, toda a informação é armazenada em uma única estrutura homogênea.

Os modelos semânticos são freqüentemente chamados de modelos orientados para objetos, em oposição aos modelos mais tradicionais de bases de dados, chamados de modelos orientados para registros. A razão para isso é que, enquanto nos modelos tradicionais as informações são armazenadas em estruturas (arquivos, relações) homogêneas, compostas de atributos básicos, nos modelos semânticos as estruturas podem ser heterogêneas, contendo instâncias de objetos que não mantêm entre si uma estrutura que pode ser generalizada em uma tabela, e além de poderem ser compostas de atributos básicos, podem também ser compostas de outras estruturas.

Além disso, além das inter-relações de composição entre objetos, um modelo semântico em geral suporta a especificação de inter-relações de generalização entre objetos, no sentido de que um determinado objeto pode constituir-se em uma generalização de um conjunto de outros.

II.4.4.2. - O Construtor de Bases de Dados Semânticas - SEDACO.

Um exemplo de uma ferramenta computacional que usa um modelo semântico é o "Semantic Database Constructor - SEDACO [FAR_85]. O modelo sobre o qual essa ferramenta atua descreve uma base de dados como uma coleção de objetos, cada um identificado por um nome específico. Conjuntos de objetos semelhantes são agrupados em tipos de objetos, que também são identificados por seus nomes, e são interconectados através de atributos.

Tipos podem ser subdivididos em hierarquias de subtipos, que têm seus objetos determinados através de funções definidas sobre os atributos do tipo mestre. Várias restrições podem ser impostas sobre os valores dos atributos, através da especificação de características para cada atributo. Assim um atributo pode ser:

- Primário: existe apenas um objeto de determinado tipo com um determinado valor para esse atributo;
- Uni-Valorizado: Somente um valor é aceito para esse atributo;
- Multi-valorizado: O atributo pode assumir vários valores simultaneamente para o mesmo objeto;
- Ordenado: indica que existe uma ordem nos valores assumidos pelos atributos de cada objeto de um determinado tipo.

SEDACO é uma ferramenta efetivamente implementada, dispondo de operadores para a definição de esquema de dados, e operadores para manipulação de dados, tendo sido totalmente desenvolvido apoiado no esquema semântico apresentado, e não como uma extensão de algum Sistema de Gerenciamento de Base de Dados tradicional.

II.4.4.3. - O Modelo de Associação Semântica - SAM*.

O Modelo de Associação Semântica (Semantic Association Model - SAM*) desenvolvido por Stanley Su [WSU_86] é um modelo conceitual para a descrição de esquemas de dados voltado para a descrição de modelos de Bases de dados para aplicações estatísticas e científicas, aprimorado posteriormente para englobar aplicações de PAC/FAC e Fabricação Computadorizada Integrada.

Trata-se de um modelo desenvolvido para permitir a representação e a análise das especificações e das propriedades semânticas para bases de dados usadas em aplicações de PAC/FAC e Fabricação integrada. O modelo consiste em um conjunto de regras para obter a modelagem de situações do mundo real que são então expressas através de diagramas de dados. Segundo o autor, o SAM* pode ser usado para identificar e representar situações em que ocorrem:

- Tipos de Dados Complexos: para manipular dados tais como vetores e matrizes, além dos tipos de dados comumente manipulados por bases de dados convencionais;
- Relacionamentos Temporais, Posicionais e Procedimentais: para permitir a representação do significado dos relacionamentos em qualquer nível de abstração de dados;
- Hierarquias de Estruturas de Dados: para permitir que objetos e eventos sejam especificados através da composição de diferentes estruturas de dados;
- Definição de Objetos de Dados Recursivamente: para permitir que objetos sejam definidos a partir de estruturas semelhantes a sua própria estrutura;

- **Objetos Complexos:** para permitir que um mesmo tipo de objeto possa representar ocorrências heterogêneas de dados daquele tipo;
- **Atributos que Indicam Valores em Domínios Alternativos:** para permitir que atributos possam ser representados usando diferentes notações;
- **Dados Particionados e Duplicados:** para permitir o uso de bases de dados distribuídas;
- **Identificação e Especificação de Sub-Bases de Dados:** para permitir que alternativas de projeto sejam criadas e testadas;
- **Versões:** para permitir a armazenagem de diferentes versões de um mesmo projeto ou produto;
- **Atributos de Categorização e Sumário:** para permitir operações estatísticas sobre os dados da Base de Dados.

Usando o SAM* uma base de dados pode ser modelada através de uma rede de objetos, denominados "Conceitos", inter-relacionados através de "Associações". Existem dois tipos de Conceitos: Atômicos e Não-Atômicos. Conceitos Atômicos são aqueles que não podem ser decompostos em outros, e além disso apresentam um significado por si só, tal como a peça de projeto 10-045. Os conceitos Não-Atômicos são obtidos através da associação de conceitos Atômicos e Não-Atômicos.

Um esquema de dados produzido usando o SAM* é uma representação gráfica dos dados. Os conceitos Não-Atômicos são modelados através de círculos onde é inscrita uma letra indicando o tipo de associação que forma aquele conceito. O nome do conceito é escrito ao lado do círculo. As associações são

representadas através de setas que indicam para cada conceito formado quais são os conceitos que o formam. Conceitos usados para identificação de cada conceito são indicados através de três cortes na seta que indica aquele conceito. Os conceitos atômicos são representados somente através dos conceitos não atômicos do qual os conceitos atômicos constituem os domínios de abrangência.

O SAM* admite a representação de sete tipos diferentes de associações. Cada tipo de associação tem algumas restrições estruturais e propriedades semânticas pré-definidas, que definem a forma como os conceitos componentes se inter-relacionam, e devem ser entendidos pelo Sistema de Gerenciamento de Bases de Dados que usar o SAM*. Os sete tipos de associações que pode-se modelar com o SAM* são os seguintes:

- Associação de Participação ("Membership association"): corresponde ao agrupamento de um conjunto homogêneo de conceitos atômicos, formando um domínio de abrangência. É representado pela letra "M";
- Associação de Agregação: um conceito pode ser a composição de seus atributos ou características. A associação de agregação corresponde à formação de um conceito composto de vários outros conceitos de forma homogênea, em uma maneira semelhante ao significado de abstração de agregação de Smith & Smith. É representado pela letra "A";
- Associação de Interação: dois conceitos formados por associações de agregação podem interligar-se para identificar um outro conceito que depende da relação dos dois conceitos relacionados, correspondendo ao relacionamento com atributos de relacionamento do Modelo Entidade-Relacionamento. É representado pela letra "I";

- Associação de Generalização: corresponde à generalização de tipos de conceitos, de uma maneira semelhante à abstração de generalização de Smith & Smith. É representado pela letra "G";
- Associação de Composição: corresponde à associação entre conceitos diferentes, que podem compôr um outro objeto. Corresponde ao tipo de relacionamento "ser parte de", que pode compôr um conceito mesmo a partir de outros conceitos diferentes. É representado pela letra "C";
- Associação de Produto-Geral (Cross-Product Association): corresponde ao conjunto de todas as ocorrências de atributos dos conceitos formados. Cada conceito é portanto um conjunto, passível de operações estatísticas. É representado pela letra "X";
- Associação de Sumarização: corresponde ao conjunto de atributos que sumarizam cada conjunto de atributos de um conceito formado por uma associação de Produto-Geral. É representado pela letra "S".

II.4.4.4. - O Modelo de Dados Semântico - SDM.

O Modelo de Dados Semânticos (Semantic Data Model - SDM) é constituído por um formalismo para descrever um esquema de dados para bases de dados, que vem sendo desenvolvido por Hammer & McLeod [HAM_81]. Os autores argumentam que as estruturas de dados que podem ser representadas pelos modelos tradicionais não suportam o desenvolvimento e a manutenção de Bases de Dados complexas, por terem uma capacidade muito limitada de representação semân-

tica, e propõem o Modelo de Dados Semântico como uma alternativa para a modelagem de dados, capaz de representar uma grande quantidade de informações semânticas, buscando satisfazer vários critérios:

- Ser um modelo apoiado em construções estruturais com alto conteúdo semântico, em oposição aos modelos tradicionais, apoiados em registros, o que segundo os autores causa uma modelagem "plana", que não consegue representar os detalhes semânticos adequadamente;
- Permitir a descrição de modelos relativistas, em que uma mesma informação do mundo real possa ser representada de diferentes maneiras, de acordo com a necessidade de cada aplicação em particular. Para isso, é proposto que o modelo tenha uma coleção de construções bastante rica, que possa dar flexibilidade de escolha de representação e capacidade de redundância lógica na informação expressa pelo esquema de dados;
- Ser um modelo que permita a definição de esquemas baseados em entidades abstratas, tais como coleções de objetos, relacionamentos entre objetos, e interconexões estruturais entre coleções de objetos. Os autores indicam que todas as entidades representadas em um esquema tenham seus próprios nomes, os quais são considerados como distintos da entidade em si.

Os princípios que norteiam a modelagem de uma situação do mundo real usando o formalismo proposto pelo SDM, apoiam-se nos conceitos de "entidade", que correspondem aos objetos que existem no ambiente a ser modelado, sejam eles reais ou abstratos, e

"classes", que são coleções de entidades que formam conjuntos de entidades que tenham um significado, e aos quais pode-se dar nomes. Toda classe tem uma coleção de atributos, que descrevem as entidades que compõem essa classe. Os atributos podem ser de dois tipos: Atributos de Membros, que são associados a cada entidade que compõe a classe; e Atributos de Classe, que descrevem propriedades da classe tomada como um todo.

As classes são divididas em: classes básicas - aquelas que podem ser definidas de forma independente de qualquer outra do esquema de dados; e classes não básicas - aquelas que são definidas em termos de uma ou mais outras classes.

As classes não básicas estão associadas através de conexões inter-classes ("inter-class connections"). Essas conexões inter-classes são divididas em Conexões de Sub-Classes, e Conexões de Agrupamento.

As Conexões de Sub-Classes indicam que as entidades que são membros de uma entidade podem também fazer parte de Subclasses dessa classe. Assim, uma subclasse é também uma classe que contém algumas, mas não necessariamente todas as entidades da classe básica. Uma conexão de Sub-Classe é especificada através de um predicado satisfazendo uma condição definida sobre o valor de um atributo de membro da classe básica.

As conexões de agrupamento permitem expressar classes cujas entidades membro que as compõem são também outras classes.

Os autores chamam a atenção para o fato de que cada uma das classes não básicas são definidas através de apenas uma única conexão interclasse, apesar de que freqüentemente mais de uma conexão poderia expressar a mesma classe. Isso é proibido sob a alegação de que mais de uma conexão interclasse seria redundante

e traria uma complexidade adicional desnecessária ao esquema.

Os relacionamentos que existem entre as entidades de cada classes são feitas através dos valores dos atributos que compõem cada classe, de uma maneira semelhante ao que ocorre no modelo relacional. Assim, especificando-se como valor de um atributo em uma classe o mesmo valor de um atributo de identificação em outra classe, o relacionamento fica estabelecido e pode ser usado para acessar outras entidades em outras classes.

E' possível indicar um relacionamento: entre apenas duas classes, e nesse caso para todo relacionamento é indicado um relacionamento inverso; e relacionamentos envolvendo várias classes, sendo que nesse caso um mesmo atributo é usado para comparar a ocorrência de seus valores em várias classes simultaneamente.

Segundo os autores, o SDM não pretende esgotar todas as possibilidades de representação de semântica de dados em esquemas apropriados para bases de dados, mas prover o que os autores apresentam como relação 80-20, ou seja, atender 80% das construções necessárias à representação das bases de dados com apenas 20% dos recursos disponíveis, e manter o SDM, como um formalismo para a representação de esquemas de dados, aberto para futuras extensões que se mostrarem necessárias e adequadas.

II.4.4.5. - O Modelo Entidade-Categoria-Relacionamento.

O Modelo Entidade-Categoria-Relacionamento (ME-C-R) é um modelo semântico concebido como uma extensão do ME-R [NAV_86a] para incorporar o conceito de abstração de generalização. As seguintes extensões são feitas ao ME-R:

- é introduzido o conceito de categoria para representar hierarquias de generalização e subtipos;
- são refinadas as restrições sobre relacionamentos para especificar como as entidades dos vários tipos e subtipos podem participar dos relacionamentos.

Uma categoria é um subconjunto de entidades do conjunto de entidades de um determinado tipo ou de outra categoria. Um conjunto de entidades é chamado de Classe de Objetos, seja o conjunto composto por entidades de um tipo ou de uma categoria. A especificação de relacionamentos no ME-C-R é feita especificando-se as classes de objetos que participam do relacionamento, e se a classe contiver outras, os objetos pertencentes às classes contidas também participam de seus relacionamentos. Os atributos que são associados a uma determinada classe também são "herdados" pelas entidades das classes nela contidas.

Para representar um esquema de dados graficamente, um novo símbolo é introduzido no Diagrama de Entidades e Relacionamentos, para indicar as Categorias, consistindo em um hexágono, que pode ser associado aos retângulos que representam entidades através de segmentos de retas cortados por um "C", indicando que a Categoria está Contida naquela entidade.

II.5. - Conceitos Adicionais Necessários ao Modelo Proposto.

O desenvolvimento do modelo proposto no capítulo III deste trabalho requer alguns conceitos adicionais aos já apresentados, que serão expostos a seguir. Esses conceitos dizem respeito principalmente a bases de dados distribuídas, uma vez que o modelo é conveniente para esse tipo de organização, e ao controle de cópias de blocos de dados da base, mantidas separadamente para efeito de desenvolvimentos paralelos de alternativas e versões de um mesmo projeto por mais de um projetista simultaneamente.

II.5.1 - Ativação Remota de Processos.

Em sistemas distribuídos, cada nodo da rede contribui realizando parte da tarefa global. Para que o sistema como um todo possa atuar coordenadamente, dados e sinais são trocados entre os processos, geralmente em forma de mensagens. Para que esses vários nodos possam cooperar, eles devem estar preparados tanto para transmitir quanto para receber os dados, porém além disso, sinais de controle podem circular pela rede, e dessa maneira é possível que se estabeleça uma arquitetura e um protocolo de comunicação de tal forma que um nodo possa solicitar a outro nodo a execução do procedimento. Essa ação é denominada Ativação Remota de Processos - ARP (ou Remote Procedure Call - RPC).

O objetivo da técnica de ARP é permitir que a lógica de um sistema seja construída sem levar em conta as dificuldades inerentes à comunicação em um sistema distribuído, tal como a forma como a passagem de mensagens é feita e diferenças existentes

entre os vários nodos de uma rede. Sob o enfoque da técnica de ARP, ativar um procedimento remoto é sintaticamente visto como uma chamada de uma subrotina local. Dessa forma, a técnica de ARP consiste em um mecanismo implementado no nível lógico mais alto de uma interface para comunicação de mensagens [SHR_82].

Se uma linguagem de programação contiver primitivas de chamadas de procedimentos remotos, os programas poderão ser construídos como se estivessem operando em um sistema com apenas um processador, porém poderá rodar sem modificações em um sistema com múltiplos processadores e aproveitar os recursos de paralelismo que este oferecer. Um exemplo de linguagem que contém essas primitivas é o sistema EDEN [ALM_85].

O ambiente de processamento onde a técnica de Ativação Remota de Procedimentos toma lugar é constituída de um conjunto de nodos comunicantes, cada nodo provendo alguns serviços através de processos, que podem ser usados por outros processos locais ou remotos. Quando um processo "A" chama um processo "B", estejam ou não os dois processos no mesmo nodo, o processo "A" denomina-se "Cliente", e o processo "B" denomina-se "Servidor". Normalmente um cliente deve passar dados para seus servidores, que são processados, e a seguir recebe deles outros dados correspondentes a esse processamento.

Para efeito de controle de uma linguagem que incorpore essas primitivas, considera-se que quando um Cliente chama um Servidor, o Cliente fica "Suspenso", ao passo que o Servidor é "Ativado". Quando o Servidor termina a operação solicitada, o Cliente é "Liberado" e o Servidor se torna "Inativo". É possível a um Cliente ativar vários Servidores ao mesmo tempo, e nesse caso ele somente é Liberado quando todos os Servidores terminarem suas

operações. A sequência de operações em geral segue o seguinte algoritmo:

Cliente		Servidor
Ativa(servidor)		Ativado
Envia(Dados)	---->	Recebe(dados)
Suspenso		. . .
		processamento
Liberado		. . .
Recebe(dados)	<----	Envia(dados)
. . .		Inativo

Cada processo é considerado no sistema como uma entidade autônoma. Assim, é possível que um Servidor chame por sua vez outros servidores, tornando-se cliente daqueles. Nesse caso considera-se que o primeiro cliente é também cliente dos servidores chamados pelo seu servidor. Para referenciar-se ao processo que chamou um determinado servidor diretamente é usado o termo "Cliente Direto".

Os processos podem ser atômicos ou não atômicos. Atomicidade é uma propriedade importante para a técnica de ARP, devido ao fato que uma chamada de um procedimento pode não terminar da maneira esperada, devido a erros detectados ou ocorridos durante o processamento. A propriedade de atomicidade é apoiada em dois conceitos denominados Integralidade e Serializabilidade.

O conceito de Integralidade requer que uma operação somente possa ocorrer como um todo. Dessa forma, uma operação ou realiza

o que dela se solicita integralmente, e seu resultado torna-se disponível para todo o sistema, ou nada é feito, e tudo permanece como se a chamada não houvesse ocorrido. Esse conceito é importante quando se trata de operações sobre Bases de Dados em sistemas distribuídos, pois se porventura alguma alteração já houver sido feita na base de dados e a seguir um erro impedir que a operação fosse completada, então a Integralidade de operações requer que a alteração seja desfeita, de maneira a preservar a consistência da base de dados.

O conceito de Serializabilidade requer que o efeito de executar mais de uma operação em paralelo seja o mesmo de se efetuar essas mesmas operações em sequência. Isso significa que a ordem com que os servidores atendem aos clientes não deve afetar o resultado final do sistema. Lin e Gannon [LIN_85] apresentam um conjunto de regras semânticas que determinam quando um processo pode aceitar uma chamada de um cliente, e que são as seguintes:

- se um processo está suspenso, ele pode aceitar chamadas apenas de seus servidores (Chamadas recursivas);
- se um processo está inativo e a última vez que ele foi ativado foi através de uma chamada de um processo não-atômico, então ele pode aceitar chamadas de qualquer processo,
- se um processo está inativo e a última vez que ele foi ativado foi através de uma chamada de um processo atômico, então ele pode aceitar chamadas:
 - de qualquer de seus clientes da última chamada;
 - de qualquer processo, desde que a última chamada tenha sido bem sucedida;
 - de qualquer processo ainda não completado que tenha

direta ou indiretamente causado a última chamada.

Nessa discussão, um processo não atômico é considerado como aquele que não afeta os dados do sistema, e portanto não está sujeito à verificação dos dois conceitos discutidos. A última regra apresentada, que permite a um processo aceitar apenas chamadas de um processo ainda não completado após o atendimento de uma chamada daquele processo, é para garantir que todos os servidores envolvidos em um processo que ainda pode potencialmente ser interrompido, possam ser solicitados para desfazer qualquer modificação já efetuada no sistema, se um cliente verificar que não pode continuar uma operação já em andamento [GRA_78].

II.5.2. - Consistência e Travamento de Dados em Bases de Dados.

Em um sistema tradicional, sempre que mais de um usuário possa ter acesso simultâneo a uma base de dados, surge o problema de conflito de acesso, ou seja, se mais de um usuário pode ter acesso simultâneo a uma determinada informação, e se um desses usuários atualizar essa informação, os outros usuários passam a correr o risco de estarem se referindo à informação desatualizada. Isso ocorre, porque o acesso à uma informação da Base de Dados sempre corresponde a uma cópia dessa informação em uma área individual de cada usuário. A atualização na Base de Dados não é acompanhada de atualização das cópias locais que os usuários possam ter. O problema torna-se crítico quando mais de um usuário decide fazer atualizações ao mesmo tempo, quando então

pode haver uma perda de uma das atualizações, ou mesmo uma corrupção da estrutura da base de dados.

Para que uma Base de Dados possa ser acessada concorrentemente por vários processos, é necessário que se disponha de um mecanismo de controle de acesso, que garanta a consistência dos dados mantidos na Base de Dados. Esse mecanismo normalmente se baseia na coordenação de operações atômicas solicitadas sobre a Base de Dados. Sendo atômicas, as operações podem ser serializadas, garantindo que cada estado que se torna visível para os programas de aplicação sejam consistentes, e correspondentes aos que seriam visíveis se as operações fossem efetivamente realizadas em alguma ordem seqüencial.

As operações que se realizam na base de dados são consideradas como compostas de ações não decomponíveis e atômicas, solicitadas sobre a base de dados. Essas ações são operações sobre os dados da base, tal como escrita ou consulta de registros ou campos. Ressalte-se que o termo ação atômica significa que o resultado de uma ação é visto sempre integralmente e nunca parcialmente, e além disso se duas ações forem solicitadas simultaneamente, o resultado é o mesmo de se forem solicitadas em alguma ordem seqüencial.

Não se pode assumir que o resultado de uma ação sobre uma base de dados que esteja consistente, sempre leve a base de dados a um outro estado consistente. Por exemplo, a escrita de um novo registro pode temporariamente levar a um estado inconsistente, se esse registro faz parte de uma tabela para a qual existe uma operação estatística associada, tal como total ou média. Para que se possa garantir a consistência em determinados pontos no tempo durante a execução de aplicações que manipulam a base de dados, é

necessário que seqüências de ações sejam agrupadas, sendo que cada grupo, denominados "transações", levam a base de dados de um estado consistente para um outro estado também consistente. Assim, a consistência que não pode ser assegurada durante a execução de uma transação, é garantida ao final desta, posto que o início da transação tenha encontrado a base consistente.

Ao passo que uma ação é constituída de um acesso sempre rápido à base de dados, uma transação pode durar muito tempo. No caso específico de uma base de dados para projeto, uma transação pode chegar a durar muitos dias. Torna-se então imperioso que as ações de várias transações possam ser encadeadas, fazendo com que muitas transações evoluam em paralelo, ao mesmo tempo que cada ação sempre veja a base de dados sem nenhuma das inconsistências introduzidas pelas demais transações.

Para se conseguir isso, é necessário que cada transação requisiite para si uma parte dos dados da base, e efetue sobre eles uma "trava" que impede que outras transações tenham acesso a eles. Isso introduz um conjunto de novas operações associadas ao protocolo de travamento dos dados, notadamente as operações de travar um dado, verificar se um dado está travado, e destravar um dado.

Uma operação de travamento de dados pode ser aceita ou não. Caso seja solicitado o travamento de um dado que já esteja travado por outra transação, então o novo pedido é rejeitado. É critério da implementação do sistema em apreço, a decisão se a rejeição de um pedido de trava de dados suspende a transação até a liberação do dado solicitado, ou se a negação da operação é notificada para a transação, que então decide que ação tomar.

É possível manter em um sistema mais de um tipo de trava.

exclusivamente em memória, o tamanho do grânulo deve ser bastante grande, existindo apenas da ordem de 50 a 200 grânulos que podem ser individualmente travados em uma base de dados [RIE_77]. Usando um número maior de grânulos para manter a eficiência de processamento, a estrutura deve então necessariamente estar na memória principal, o que novamente por si só, já limita o número de grânulos que podem ser controlados.

Uma opção de escolha para o tamanho do grânulo, é permitir que esse tamanho seja determinado pelo próprio programa de aplicação, através da especificação de um predicado que determina quais dados devem ser travados. Feita essa diferença, passa então a se denominar "travamento físico" ao travamento de grânulos de tamanho pré-determinados, tais como a trava de um registro ou de um arquivo, e a se denominar "travamento por predicado" a trava de dados que dependem de uma condição especificada juntamente com a solicitação de trava, tal como por exemplo o travamento de todos os registros de um arquivo que tenham os valores de um determinado campo dentro de uma faixa de valores.

O travamento por predicado apresenta uma série de vantagens. A principal aproveita o fato que duas aplicações em geral estarão tratando de aspectos distintos da base de dados, e portanto logicamente acessando dados diferentes. Usando travamento físico e um grânulo razoavelmente grande, é também grande a probabilidade de uma transação bloquear o acesso a dados que ela mesma não usa, mas que são necessários para uma outra transação. O travamento por predicado diminui muito a probabilidade que isso ocorra.

No entanto, dependendo do uso e da forma como um Sistema de Gerenciamento de Bases de Dados é implementado, o travamento por

Um parâmetro que deve ser decidido para se implementar um mecanismo de trava de dados, é a quantidade de dados que uma operação de travamento afeta. A escolha de um valor pré-determinado invariavelmente leva a uma solução de compromisso, pois por um lado, quanto maior a quantidade de dados travados, menor o número de operações de travamento e destravamento que cada transação deve solicitar, mas também quanto maior a quantidade de dados maior é a probabilidade de um dado não necessário para a transação ser travado desnecessariamente.

Além disso, o sistema deve manter uma estrutura de dados que indique se um dado está ou não travado, mantendo um registro de estado para cada bloco de dados que possa ser travado. Essa estrutura será tanto maior, e mais tempo será consumido atualizando-a, quanto menor for a quantidade de dados travados em cada solicitação.

Por outro lado, se a quantidade de dados travados em cada solicitação for muito grande, pode ocorrer que a maior parte das transações não possam ser executadas, por não terem acesso aos dados de que precisam.

As alternativas de tamanho do bloco de dados travados em cada operação, denominados grânulos de travamento, variam desde campos de um arquivo, registros específicos, arquivos inteiros, até a própria base de dados, quando então seria caracterizado um sistema monousuário.

Tem sido estudado quais seriam as melhores alternativas para a escolha desse tamanho, porém o valor ideal é extremamente sensível à forma como uma base de dados é implementada e usada [RIE_79]. Surpreendentemente, foi determinado que, a menos que toda a estrutura de controle de acesso seja mantida

exclusivamente em memória, o tamanho do grânulo deve ser bastante grande, existindo apenas da ordem de 50 a 200 grânulos que podem ser individualmente travados em uma base de dados [RIE_77]. Usando um número maior de grânulos para manter a eficiência de processamento, a estrutura deve então necessariamente estar na memória principal, o que novamente por si só, já limita o número de grânulos que podem ser controlados.

Uma opção de escolha para o tamanho do grânulo, é permitir que esse tamanho seja determinado pelo próprio programa de aplicação, através da especificação de um predicado que determina quais dados devem ser travados. Feita essa diferença, passa então a se denominar "travamento físico" ao travamento de grânulos de tamanho pré-determinados, tais como a trava de um registro ou de um arquivo, e a se denominar "travamento por predicado" a trava de dados que dependem de uma condição especificada juntamente com a solicitação de trava, tal como por exemplo o travamento de todos os registros de um arquivo que tenham os valores de um determinado campo dentro de uma faixa de valores.

O travamento por predicado apresenta uma série de vantagens. A principal aproveita o fato que duas aplicações em geral estarão tratando de aspectos distintos da base de dados, e portanto logicamente acessando dados diferentes. Usando travamento físico e um grânulo razoavelmente grande, é também grande a probabilidade de uma transação bloquear o acesso a dados que ela mesma não usa, mas que são necessários para uma outra transação. O travamento por predicado diminui muito a probabilidade que isso ocorra.

No entanto, dependendo do uso e da forma como um Sistema de Gerenciamento de Bases de Dados é implementado, o travamento por

predicado pode implicar que um grande número de testes deva ser feito antes de autorizar qualquer ação na base de dados, o que diminui a eficiência do sistema.

Devido a essa última consideração, os sistemas de Bases de Dados comerciais em geral usam o travamento físico, ficando o travamento por predicado em geral restrito a utilitários de interfaceamento com sistemas de bases de dados que adotem o travamento através de grânulos bastante pequenos, da ordem de grandeza de um registro.

II.5.3. - Gerenciamento de Configuração de Software.

A disciplina chamada Gerenciamento de Configuração de Software (GCS), propõe-se a estudar e apresentar regras que permitam a um gerente controlar o desenvolvimento e manutenção de um sistema de software. Trata-se de uma disciplina bastante recente [BER_84], surgida da necessidade de se controlar a construção de sistemas complexos de software, envolvendo muitas pessoas, ou um longo tempo de desenvolvimento e manutenção. Seus conceitos fundamentam-se no gerenciamento de configuração tradicional, aplicado a sistemas que são constituídos principalmente por software.

No contexto da disciplina de GCS, o termo "Software" não se refere apenas a programas, mas envolve também todos os documentos, relatórios, listagens, etc. associados. Dessa forma, procura controlar as versões e o desenvolvimento tanto de programas quanto de toda a documentação que os acompanha.

A técnica básica para o GCS consiste em estabelecer pontos, discretos no tempo, no sistema a ser gerenciado, com o propósito de sistematicamente controlar as mudanças na configuração, e permitir manter um acompanhamento e integridade da configuração durante todo o seu ciclo de vida. No instante que o sistema atinge um desses pontos ele é congelado, e passa a ser denominado uma "Configuração Básica" (Baseline), que é um ponto de referência no desenvolvimento do sistema, e representa o produto final de uma fase do seu ciclo de vida.

Da mesma forma que no domínio do tempo, um sistema também é dividido logicamente em partes em que se tenha interesse gerenciar, as quais são denominadas "Item de Configuração de Software" - ICS. Em um sistema de software podem ser considerados como ICS documentos, rotinas, bibliotecas, programas, subsistemas e o próprio sistema. Um ICS é então um componente do sistema definido formalmente para acompanhamento e controle durante o desenvolvimento e a manutenção do sistema de software.

Durante o desenvolvimento do sistema, um ICS pode ser decomposto em outros, gerando outros ICSs, e também podem sofrer alterações, gerando versões dos ICSs originais. Algumas dessas versões podem ser feitas com o intuito de obter variantes do sistema proposto, caracterizando versões diferentes dos ICSs originais, que são mantidos durante todo o desenvolvimento do sistema, gerando também versões diferentes das Configurações Básicas. As Configurações Básicas são constituídas por um conjunto de ICSs, sendo que esse conjunto em geral apresenta uma estrutura hierárquica de ICSs. Conforme o desenvolvimento do sistema avança, o refinamento dos ICSs e conseqüentemente o número de níveis dessa hierarquia aumentam.

Sempre que uma versão de um ICS é aprovada pela função de controle de configuração, ela passa também a ser controlada pelo Gerenciamento de Configuração como um novo ICS, e as Configurações Básicas em que ela é envolvida passam também a constituir-se em novas versões de Configurações Básicas.

A tarefa de gerenciar o desenvolvimento e manutenção de um sistema de software usando os conceitos do GCS é desmembrada em quatro funções:

- Identificação da Configuração de Software, que tem por objetivo a identificação e especificação de requisitos técnicos de elementos de software que requeiram atenção individual.
- Controle de Configuração de Software, que tem por objetivo controlar a autorização e a execução de alterações no sistema.
- Auditoria de Configuração de Software, que consiste na execução controlada de uma série de testes de completeza, validade e consistência sobre um conjunto de ICSs. O resultado de uma função de auditoria bem sucedida tem como resultado uma Configuração Básica do sistema.
- Administração de estado, que mantém um histórico da evolução do sistema ao longo de seu ciclo de vida, bem como seu estado corrente de desenvolvimento.

A identificação de um item de configuração de software torna-se conhecida do gerente de configuração [TRA_85] através de:

- definição de ICS;
- Identificação de Requisitos Técnicos de um ICS;

- Atribuição de Características ao ICS e a seus Componentes.

A Definição de um ICS corresponde à criação de um documento ou uma ficha onde são anotados o nome e descrição geral desse ICS, juntamente com outras informações próprias do projeto e fase de desenvolvimento pertinentes.

A Identificação de Requisitos Técnicos corresponde à especificação dos dados de como o ICS deve ser desenvolvido, quais os compromissos de projeto que devem ser assumidos, informações que indiquem como o ICS é feito, o que contém, os conceitos e ferramentas que necessita para seu desenvolvimento, em suma, descreve o ICS e fornece os dados necessários para o desenvolvimento do ICS, sem que importe a forma como esse ICS será usado em um sistema.

A Atribuição de Características à um ICS corresponde à especificação de parâmetros e dados do ICS que são necessários ao uso do ICS pelas outras componentes do sistema. As características de um ICS devem ser o mais possível independentes da forma como o ICS é feito internamente, e deve conter todas as informações necessárias para seu uso, principalmente em termos de dados de interfaceamento. Além disso, deve incorporar todas as informações que permitam a localização de todas as ocorrências desse ICS em um sistema, com o objetivo de identificar todos os pontos do sistemas que devem ser novamente testados, se porventura ocorrer alguma alteração no ICS.

A disciplina de Gerenciamento de Configuração de Software é derivada das técnicas de Gerenciamento de Configuração aplicadas a outras áreas. No entanto, o Software, tendo algumas

características próprias, tais como a não tangibilidade, e a possibilidade de modificação muito mais fácil do que para as outras coisas em geral submetidas a gerenciamento de configuração, demanda um controle de configuração com características também próprias.

A utilização que se pretende fazer desses conceitos neste trabalho, correspondem ao Gerenciamento de descrições, projetos e produção de sistemas de engenharia em geral, seja este um sistema de software ou não. No entanto, um projeto armazenado em uma Base de Dados em um computador apresenta as características a serem controladas muito mais próximas de um sistema de software do que de qualquer outro para o qual o Gerenciamento de Configuração seja aplicado. Esse é o motivo pelo qual o Gerenciamento de Configuração de Software e não um outro qualquer foi apresentado.

Os conceitos usados no Gerenciamento de Configuração de Software, são quase os mesmos necessários para um Gerenciamento de Configuração de Projetos - GCP, e as diferenças mais marcantes serão aqui discutidas.

Uma extensão necessária ao GCS, para que ele possa ser usado para o gerenciamento de configuração de projetos em geral, vem do fato de que muitas versões de um projeto podem existir simultaneamente, e essa característica traz a necessidade de se poder gerenciar propostas de alternativas para determinadas partes de um projeto. O GCS a princípio não trata de alternativas, mas esse conceito pode ser incorporado com facilidade.

Katz em [KAT_84] caracteriza "Versão" e "Alternativa" de projetos, de uma maneira que pode ser incorporada às técnicas de GCS. Considera-se que um projeto está inicialmente "em desenvol-

vimento", e que sobre esse projeto os projetistas efetuam mudanças e incorporação de dados até que ele esteja pronto para produção, quando então ele passa a ser considerado Efetivo. Esse estado corresponde a uma Configuração Básica da última fase de desenvolvimento de um produto.

Em um projeto que esteja em desenvolvimento, os projetistas podem experimentar alternativas, usando para isso uma cópia do projeto no estado atual em que se encontra, sem que com isso precisem introduzir mudanças na versão em desenvolvimento. Cada cópia de um projeto em desenvolvimento, com as respectivas alterações experimentadas, é uma "alternativa" do projeto, que pode posteriormente substituir o projeto em desenvolvimento, pode ser unida com outras alternativas para substituir o projeto em desenvolvimento, ou pode simplesmente ser armazenada para referência futura. A cópia das informações correspondentes ao projeto oficial é feita através da criação de novos Itens de Configuração de Projetos (ICP) para cada ICP do projeto original que deva ser modificado.

Quando um projeto em desenvolvimento torna-se efetivo, ele deve ser submetido a testes de aceitação, a critério da organização que o desenvolve. Se aprovado, ele estará pronto para entrar em produção e ser distribuído, ou para ser submetido a fases posteriores de desenvolvimento, passando a constituir-se em uma Configuração Básica de Projeto. Nesse caso ele se torna uma "Versão" do projeto para aquela fase de desenvolvimento, e como tal ele se torna "Congelado", impedindo futuras modificações. Uma versão pode ser copiada e descongelada, permitindo posteriores alterações no projeto, que uma vez terminadas serão tornadas efetivas, gerando novas versões do projeto.

As alternativas dos ICPs originais não irão constituir-se em novas Configurações Básicas, uma vez que, se uma dada alternativa for aprovada para gerenciamento ela passa a ser uma nova versão, e a partir daí tratada apenas como tal.

O controle de quando uma versão torna-se congelada, ou quando se permitem mudanças em uma versão já efetivada, é realizado pelo processo de Controle de Alterações do GCP. Para que se possa exercer o controle e manter um histórico de todo o desenvolvimento de um sistema, uma versão que tenha sido efetivada não poderá mais ser modificada. Dessa forma, um descongelamento é feito tirando-se uma cópia, que será modificada. Note-se que com esse enfoque, um projeto pode ter muitas versões e muitas alternativas, sendo que todas as versões produzidas serão consideradas não destrutíveis, devendo ser armazenadas por tempo indefinido.

II.6. - Conclusões.

Neste capítulo foram apresentados os conceitos básicos necessários ao desenvolvimento de um modelo para uma base de dados que satisfaça as necessidades de aplicações científicas e as necessidades de desenvolvimento e produção de produtos de engenharia, em todas as suas variadas formas, desde a produção de software até a produção de circuitos integrados e engenharia mecânica. Esses conceitos envolvem desde os modelos ditos tradicionais de bases de dados, até os modelos mais recentes, que permitem uma representação semântica muito mais intensa, chamados

por isso modelos semânticos. Foram também expostos conceitos que permitem o desenvolvimento de características de bases de dados distribuídas intrínsecas ao modelo, e que serão exploradas no capítulo seguinte.

Além disso, foi efetuado um levantamento das necessidades que uma base de dados construída especialmente para o apoio a aplicações de engenharia deve satisfazer. Isso foi feito separando-se as tarefas mais gerais executadas por um Sistema de Gerenciamento de Dados para Engenharia, e analisando-se as necessidades de cada tarefa individualmente. Desse levantamento, foi elaborado um conjunto de 10 itens principais que devem ser satisfeitos. Esse conjunto será novamente discutido ao final do Capítulo IV, como uma constatação de que o modelo proposto no Capítulo III atende as necessidades de um Sistema de Gerenciamento de Dados para Engenharia.

C A P Í T U L O I I I

O MODELO DE REPRESENTAÇÃO DE OBJETOS

III.1. - Introdução.

O Modelo Entidade-Relacionamento (ME-R) foi concebido para suportar a representação de dados em sistemas onde um Modelo de Rede ou um Modelo Relacional não pudesse expressar todo o conteúdo semântico necessário, e é exatamente essa a justificativa usada em [CHE_76] para que o Modelo Entidade-Relacionamento seja apresentado.

No contexto deste trabalho, o termo "semântica" será usado para indicar as propriedades estruturais e as restrições de integridade dos dados. Assim, para se fazer o modelo de dados de uma aplicação em engenharia, é necessário que se identifiquem as propriedades semânticas da aplicação, e que estas sejam colocadas de forma explícita na especificação do modelo de dados, e que sejam reconhecidas e seguidas pelo sistema de gerenciamento de dados que irá suportar tal aplicação.

Desde a sua formalização em 1976, o Modelo Entidade-Relacionamento tem recebido atenção crescente em aplicações em que a necessidade de maior conteúdo semântico causa restrições ao uso dos demais modelos, e o seu uso tem se tornado uma constante em sistemas de representação de dados de engenharia. Essa

tendência vem mostrar que a proposta inicial estava na direção certa, e que o modelo é adequado para as aplicações a que se propôs apoiar.

No entanto, nas aplicações não convencionais em que se procura usar esse modelo, pode-se notar que em geral são feitas propostas de modificações ou extensões ao modelo original, seja para melhorar seu desempenho em alguma característica especial, seja para lhe incorporar características necessárias mas não cobertas pelo modelo original. Pode-se considerar o Modelo Entidade-Relacionamento, tal como foi originalmente proposto e é formalmente definido, como um eixo em torno do qual as implementações existentes incorporam extensões ou efetuam restrições.

Visto dessa forma, o modelo que será apresentado a seguir, é inicialmente apresentado através de extensões ao modelo original. Essas extensões se justificam pelo fato de que as necessidades de um Sistema de Gerenciamento de Dados para Engenharia exigem que seja representado um conteúdo semântico ainda maior do que é possível no ME-R original.

Os modelos semânticos mais recentes apresentam cada um muitas características que são usadas também no modelo a ser apresentado, no entanto a apresentação é feita como uma extensão ao Modelo Entidade-Relacionamento principalmente por dois motivos: primeiro por que o ME-R é mais conhecido, estando mais "estabilizado" do que os modelos semânticos mais recentes; segundo por que nenhum dos modelos semânticos surgidos na literatura serve realmente de Base para o modelo a ser apresentado, ou por que o enfoque tomado é diferente, ou por que contêm estruturas que invalidam determinados desenvolvimentos que se pretende fazer. Sendo um modelo mais básico, o ME-R apresenta menor quantidade de

construções específicas que polarizem o desenvolvimento do modelo apresentado.

O modelo proposto será denominado Modelo de Representação de Objetos (MRO), a palavra "Objeto" aqui significando "o que é conhecido ou representado, em oposição ao ato de conhecer ou representar; tudo o que apresenta à percepção um caráter fixo ou estável" [FER-75]. Esse nome foi escolhido, em oposição ao termo [FER_75] "Entidade", porque o modelo procura representar objetos (entes, eventos, coisas, conceitos), tratando cada um de forma individual, mantendo na representação de cada objeto todas as informações pertinentes, sem que se procure estabelecer, a priori, padrões para a representação de conjuntos de objetos, tal como é o caso do Modelo Entidade-Relacionamento, onde a representação das informações é feita na realidade através das relações de dados das entidades e dos seus relacionamentos, especificados de forma imutável no esquema de dados, depois da inicialização da base de dados.

As principais modificações que o Modelo de Representação de Objetos apresenta em relação ao Modelo Entidade-Relacionamento podem ser sintetizadas no seguinte conjunto:

- Adoção de uma Estrutura de Dados de Acesso à informação que garanta flexibilidade para a definição e posteriores modificações ou redefinição dos Esquemas de Dados, e que permita o uso do modelo em um Meta-Sistema de Gerenciamento de Dados;
- Maior especificidade em características de atributos, de forma a que algumas características intrínsecas a alguns tipos de atributos se transfiram para a semântica desses atributos;

- Possibilidade de passar ao Sistema de Gerenciamento de Dados o escopo de abrangência dos identificadores de objetos usados em um dado estágio de interação do sistema com um usuário, para refletir no sistema uma maior especificidade em um dado momento, especificidade essa normalmente admitida pelo usuário;
- Possibilidade de indicar com maior coesão a identificação de objetos através de relacionamentos, para permitir que atributos de um ou mais objetos ascendentes sejam "herdados" por objetos descendentes, identificados pelos relacionamentos que envolvem os objetos ascendentes.

Nas seções III.2 a III.6 cada um desses tópicos são analisados em detalhe, introduzindo-se progressivamente os conceitos que constituem o MRO. Assim, na seção III.2 é apresentado o conceito de Meta-Sistemas de Gerenciamento de dados, e como é possível permitir alterações no Esquema de Dados de uma base já inicializada. Na Seção III.3 o Modelo de Representação de Objetos tem os seus principais conceitos apresentados, juntamente com a estrutura lógica de acesso à informação. Na seção III.4 é apresentado o conceito de Característica de Atributos, que permite aumentar a semântica das operações realizadas pelo SGBD no tratamento de atributos. Na seção III.5 apresenta-se o conceito de Classes, que permite ao SGBD acompanhar o nível de abstração e localização de interesse nas informações acessadas pelo usuário. Na seção III.6 é apresentado o conceito de Identificação de Objetos Através de Relacionamentos, o que permite ao SGBD responder às necessidades de

objetos "herdarem" atributos de seus ascendentes, bem como atender à operações que envolvam a ocorrência de instâncias de objetos armazenados na Base de Dados.

Por fim na seção III.7 analisa-se como o MRO se adequa a execução em um ambiente multiusuário, ressaltando-se as vantagens advindas de sua execução em um ambiente de multiprocessamento.

III.2. - Definição e Modificações em Esquemas de Dados no Modelo de Representação de Objetos.

Pode-se entender um Meta-Sistema como um sistema que usa uma estrutura semelhante à sua própria estrutura para definir-se a si mesmo. Por exemplo, uma linguagem que ensine, ou especifique uma linguagem será uma Meta-Linguagem; um programa de computador que possa ser definido, feito ou melhorado quando se usa esse mesmo programa será um Meta-Programa.

Em um Meta-Sistema de Gerenciamento de Dados, o sistema deve poder aceitar a Definição do Esquema de Dados da mesma maneira que aceita os dados, pois o próprio esquema de dados deve ser encarado como um conjunto de dados que deve ser gerenciado. Tal enfoque, entre outras coisas implica em que as operações de definição e gerenciamento do esquema de dados devem ser suficientemente flexíveis para permitir alterações no esquema de dados para que se possa admitir que dados não previstos, ou diferentes da especificação inicial, possam vir a ser colocados na Base de Dados. No caso de uma aplicação que use um (Meta) Sistema de Gerenciamento de Dados, esse enfoque poderia admitir que o Esque-

ma de Dados da aplicação fosse alterado sem que a aplicação tivesse que ser modificada.

III.2.1. - Alteração do Esquema de Dados no ME-R Depois da Base de Dados já Inicializada

O Modelo Entidade-Relacionamento admite que qualquer entidade do domínio de abrangência de uma aplicação seja descrito, bem como todos os Relacionamentos que existam entre quaisquer dessas entidades. No entanto o enfoque tradicional de se construir a estrutura de acesso dos dados em uma Base de Dados que use o ME-R prescreve que as entidades semelhantes devem ser mantidas juntas em um arquivo próprio, envolvendo todos os atributos que sejam chave de acesso de cada entidade, juntamente com todos os demais atributos, e além disso, que cada relacionamento também seja mantido em um arquivo próprio, contendo como chave de acesso uma composição dos atributos que identificam cada uma das entidades que estão envolvidas no relacionamento (no mínimo duas).

Uma diferença fundamental que existe entre o Modelo Entidade-Relacionamento e o modelo Relacional, consiste justamente no fato de que, no primeiro o relacionamento entre duas entidades é explicitado através de um arquivo que o mantém, ao passo que no segundo um relacionamento é mantido através da existência de um atributo homônimo em cada uma das Relações.

Dessa forma, no Modelo Relacional, quando se deseja cruzar as informações de uma Relação com outra, cria-se efetivamente um

novo arquivo que contém os atributos desejados de cada uma das Relações de dados originais. Essa maneira de realizar as operações lógicas permite que se incorporem a um Esquema de Dados novos Atributos e novas Relações (denominadas Relações Virtuais), que não haviam sido inicialmente previstas.

No Modelo Entidade-Relacionamento no entanto, o cruzamento de informações é efetuado através das relações de Relacionamentos, as quais não podem ser criadas e nem mesmo modificadas depois que o Esquema original de dados já tenha sido utilizado para criar os arquivos que ele especifica.

Isso significa que no Modelo Entidade-Relacionamento uma Base de Dados não pode ter seu Esquema modificado depois da Inicialização da Base de Dados. Essa restrição inviabiliza seu uso em um Meta-sistema, visto que obriga a definição de um Esquema de Dados imutável, o que vem invalidar a existência de aplicações que possam atuar com um Esquema de Dados genérico, como seria o caso de um Meta-Sistema.

III.2.2. - O Uso do MRO como um Meta-Sistema de Gerenciamento de Dados.

Uma maneira de possibilitar alterações em uma base de dados já inicializada, para permitir o desenvolvimento de uma aplicação que requeira um Meta-Sistema, consiste em modificar a estrutura de acesso de dados do modelo Entidade-Relacionamento, fazendo com que todas as entidades e todos os relacionamentos possam ser armazenados em um único arquivo de entidades e um único arquivo

de relacionamentos. Isso significa que todos os campos de um arquivo de entidades ou relacionamentos tradicional devem ser padronizados, de maneira a que possam ser todos tratados de maneira homogênea.

Essa modificação constitui a primeira alteração que diferencia o MRD do ME-R. Logicamente, essa mudança não distingue os dois modelos, porém conceitualmente permite que o MRD possa ser usado em um Meta-Sistema de Gerenciamento de Dados, enquanto o ME-R não o permite. Por exemplo, considere-se que uma Base de Dados deve armazenar informações sobre Projetos de um Laboratório de Desenvolvimento de Sistemas Digitais. Inicialmente, considere-se que deva ser armazenada a informação referente a quais pessoas participam de cada projeto em andamento no Laboratório. A Figura 7 mostra o Esquema Lógico dessa Base de Dados.

Figura 7 - Esquema de Dados das Pessoas que participam de um Projeto.

Uma vez inicializada uma Base de Dados usando esse esquema, os dados correspondentes serão armazenados, e a estrutura interna que esses dados assumem depende do modelo de dados adotado pelo Sistema de Gerenciamento da Base de Dados. Na Figura 8 aparece a forma como os dados seriam armazenados em uma Base de Dados usando o ME-R, e na Figura 9 aparece a forma como os mesmos dados seriam armazenados em uma base de dados usando o MRD.

PESSOA	PARTICIPA	PROJETO										
Paulo José João	<table><tr><th>PESSOA</th><th>PROJETO</th></tr><tr><td>Paulo</td><td>Video Gráfico</td></tr><tr><td>José</td><td>Digitalizador</td></tr><tr><td>Paulo</td><td>Digitalizador</td></tr><tr><td>João</td><td>Video Gráfico</td></tr></table>	PESSOA	PROJETO	Paulo	Video Gráfico	José	Digitalizador	Paulo	Digitalizador	João	Video Gráfico	Digitalizador Video Gráfico
PESSOA	PROJETO											
Paulo	Video Gráfico											
José	Digitalizador											
Paulo	Digitalizador											
João	Video Gráfico											

Figura 8 - Dados armazenados em uma Base de Dados ME-R.

OBJETOS			RELACIONAMENTOS		
Cod	Nome	Tipo	Cod Orig	Cod Dest	Tipo
1	Digitalizador	Projeto	2	4	Participa
2	Paulo	Pessoa	3	1	Participa
3	José	Pessoa	2	1	Participa
4	Vídeo Gráfico	Projeto	5	4	Participa
5	João	pessoa			

Figura 9 - Dados armazenados em uma Base de Dados MRD.

Se alguma alteração se fizer necessária no esquema de dados, no Modelo Entidade-Relacionamento a alteração irá se refletir na estrutura dos arquivos já criados, ou ocasionará a criação de novos arquivos. Por exemplo, se for necessária a inclusão de um novo atributo entre os atributos já existentes em uma determinada entidade, então o arquivo que contém os dados dessa entidade deve ser mudado para permitir a armazenagem de um novo campo que contém o novo atributo. Se um novo conjunto de entidades precisar ser armazenado na Base de Dados, ou um novo relacionamento for estabelecido entre entidades já existentes, então um novo arquivo deverá ser criado.

Como exemplo, suponha-se que o esquema de dados mostrado na Figura 7 deva ser expandido para que sejam representadas as atividades que compõem cada projeto, sendo que as Atividades

deverão ser armazenadas como um novo conjunto de entidades, associadas aos Projetos através de Relacionamentos "Composto por". Na Figura 10 aparece o novo Esquema de Dados, incorporando essa extensão.

Figura 10 - Inclusão das Atividades que compõem cada projeto.

Uma vez inicializada a nova base de dados usando o ME-R, constata-se que é necessário que se criem mais dois arquivos além dos que já existiam anteriormente, para que as novas informações sejam representadas, o que pode ser visto na Figura 11.

PESSOA	PARTICIPA		ATIVIDADE
Paulo José João	Paulo José Paulo João	Video Gráfico Digitalizador Digitalizador Video Gráfico	Controlador de Video Amplificador de Video Sensor Magnético Gerador de Pulsos Interface Serial
PROJETO	COMPOSTO POR		
Digitalizador Video Gráfico	Video Gráfico Video Gráfico Digitalizador Digitalizador Video Gráfico Digitalizador	Controlador de Video Amplificador de Video Sensor Magnético Gerador de Pulsos Interface Serial Interface Serial	

Figura 11 - Novos dados armazenados em uma Base de Dados ME-R.

Uma base de dados usando o Modelo de Representação de Objetos, dispondo de apenas um arquivo para armazenar todos os obje-

tos e apenas um arquivo para representar todos os relacionamentos, não terá a sua estrutura alterada para a representação de outros tipos de objetos ou de relacionamentos, mas tão somente um aumento no número de registros que compõem cada arquivo. A Figura 12 mostra como os dados da Figura 11 ficariam armazenados em uma base de dados usando o MRO, de onde pode-se constatar que não existe nenhuma alteração na estrutura dos arquivos já existentes antes de se introduzir a modificação, como se verifica confrontando-se essa figura com a Figura 9.

OBJETOS			RELACIONAMENTOS		
Cod	Nome	Tipo	Orig	Dest	Tipo
1	Digitalizador	Projeto	2	4	Participa
2	Paulo	Pessoa	3	1	Participa
3	José	Pessoa	2	1	Participa
4	Video Gráfico	Projeto	5	4	Participa
5	João	Pessoa	4	6	Composto
6	Controlador de Video	Ativid.	4	7	Composto
7	Amplificador de Video	Ativid.	1	8	Composto
8	Sensor Magnético	Ativid.	1	9	Composto
9	Gerador de Pulsos	Ativid.	4	10	Composto
10	Interface Serial	Ativid.	1	10	Composto

Figura 12 - Novos dados armazenados em uma Base de Dados MRO.

A armazenagem de todos os objetos e todos os relacionamentos em apenas dois arquivos, obriga a que duas consequências principais sejam analisadas: a especificação de um atributo e a identificação de entidades.

A especificação de um atributo passa a ser restrita aos formatos padrão que tenham sido estabelecidos. Isso significa por exemplo que, se um determinado Esquema de Dados tenha a princípio determinado que os atributos serão apenas do tipo Inteiro-de-4-Dígitos e Cadeia-de-40-Caracteres, todos os atributos que forem

associados a qualquer entidade ou relacionamento deverão se enquadrar em um desses tipos.

Essa restrição pode a princípio parecer muito onerosa para uma aplicação, e efetivamente torna-se desaconselhável para alguns tipos de aplicações. No entanto, se for considerado que normalmente uma implementação de um Sistema de Gerenciamento de Base de Dados restringe-se a alguns poucos tipos de formato de dados padrão, pode-se perceber que essa restrição existe intrinsecamente na grande maioria das implementações existentes de Sistemas de Gerenciamento de Bases de Dados - em geral a maior quantidade de formatos de dados é provida por uma linguagem de programação associada ao SGBD. Além disso, encarando a utilização desse sistema sob o enfoque de uma aplicação que use um Meta-Sistema, nota-se que essa característica não é realmente uma restrição, pois um Meta-Sistema, por sua própria concepção de generalidade, obrigatoriamente estará restrito à manipulação de alguns poucos tipos de formatos de atributos.

A identificação de entidades por sua vez passa a necessitar de um padrão de identificação que abranja todas as entidades definidas por um Esquema de Dados de uma Base de Dados. Essa restrição envolve a necessidade de que todos os atributos usados para identificar qualquer entidade tenham um único tipo padrão de formato, e além disso, que todas as entidades sejam identificadas através de um único atributo. Tal restrição exclui o uso dessa estrutura de acesso de dados para muitas aplicações, notadamente aquelas que se beneficiam de um esquema de dados baseado em registros homogêneos de dados, mas para aplicações que necessitam de um Meta-Sistema, tal como uma base de dados para engenharia, isso é perfeitamente aceitável, visto que padroniza uma forma de

identificação que é necessária ao Meta-Sistema, exatamente da forma como essa padronização se faz necessária.

Um Sistema de Gerenciamento de Dados para Engenharia, se concebido para uso genérico e não para um determinado aspecto do projeto de uma modalidade de atividade de engenharia, será necessariamente um Meta-Sistema, uma vez que o Esquema de Dados não poderá ser totalmente definido a priori, caracterizando um projeto como um sistema em desenvolvimento que não pode ter todas as suas características e recursos necessários estabelecidos a priori, tendo portanto que ser também gerenciado pelo próprio sistema. Dessa forma, o enfoque apresentado para a estrutura de acesso de dados é conveniente para um modelo que seja adequado a um Sistema de Representação de Dados de Engenharia.

Essa mudança na estrutura de acesso à informação não traz a princípio nenhum ganho quanto ao conteúdo semântico do modelo, e nem o poderia, visto se tratar de uma mudança em um nível de abstração de dados bastante baixo. Essa mudança tem apenas o propósito de permitir suprir as necessidades apresentadas. No entanto, analisando-a em detalhe, verifica-se que esta traz uma consequência semântica que deve ser notada aqui, a qual se traduz no fato de existir para qualquer entidade um atributo específico de identificação, que tem sempre o mesmo formato. No restante deste trabalho, esse atributo será chamado resumidamente apenas de Identificador, ou Nome, de um Objeto.

Todo objeto passa então a ter um nome que é usado pelo usuário para fazer a identificação do objeto, ao passo que o sistema faz a identificação do objeto através de um código interno, que identifica univocamente cada objeto independentemente do nome dado pelo usuário. Esse identificador interno será denominado

"Código de Objeto", e somente pode ser usado pelo usuário depois que esse código houver sido formalmente designado pelo Sistema de Gerenciamento de Bases de Dados. O conceito de Código de Objeto no MRO corresponde ao conceito de designador sugerido por Codd para o Modelo RM/T, embora no RM/T seja preconizado que o código interno nunca deverá ser acessado diretamente pelo usuário.

Nas Figuras 9 e 12 os objetos aparecem sempre com o seu código já designado, e pode-se notar que os relacionamentos são indicados pelos códigos, e não pelos nomes dos objetos envolvidos em cada relacionamento.

Uma consequência direta da forma proposta de se realizar a estrutura de acesso de dados do modelo, é que as informações que precisam ser armazenadas sobre cada entidade não mais precisam ser estruturadas em relações (tabelas) de entidades semelhantes, ou seja, cada entidade é tratada de forma independente, sem que mantenha-se obrigatoriamente um padrão com qualquer outra. Essa característica é precisamente o que permite se pensar em cada coisa, evento ou conceito da aplicação como um objeto, ao invés de como uma entidade.

A estrutura de acesso de dados proposta é a que é preconizada para o Modelo de Representação de Objetos proposto. Dessa forma, tem-se a caracterização de representação das informações concernente a cada Objeto como sendo através de um conjunto de Atributos que se associam a cada objeto, sendo que a identificação de cada objeto é feita através de seu Nome, ou Identificador. Além disso, os objetos se relacionam entre si através de Relacionamentos, caracterizando assim os primeiros componentes elementares do Modelo de Representação de Objetos.

III.3. - A Estrutura de Dados de Acesso à Informação no Modelo de Representação de Objetos.

E' conveniente notar aqui que a estrutura de dados de acesso do modelo proposto preconiza a existência de um arquivo onde todas as informações sobre cada objeto são armazenadas, e um arquivo onde todas as informações sobre cada relacionamento são armazenadas.

Porém, não existirão apenas esses dois arquivos, pois para que todos os objetos sejam armazenados em um mesmo arquivo, somente os atributos comuns a todos eles poderão ser armazenados nesse arquivo. Esses atributos, que serão denominados Atributos Intrinsecos dos Objetos, podem ser considerados independentes de qualquer aplicação em que o modelo proposto seja usado, e definem então os campos do Arquivo de Objetos. Como atributos intrinsecos de um objeto podemos ter por exemplo o Nome do Objeto e a data de sua criação.

O mesmo ocorre com o Arquivo de Relacionamentos, que terá campos para armazenar apenas os dados comuns a todos os relacionamentos, que são por exemplo os identificadores de cada objeto envolvido pelo relacionamento.

Nas Figuras 9 e 12 podem ser observados alguns Atributos Intrinsecos que sempre ocorrem, tal como o Nome e o Código de cada Objeto, e os identificadores dos Objetos Origem e Destino de cada Relacionamento.

Para armazenar os atributos não intrinsecos, tanto de objetos como de relacionamentos, será necessário um arquivo para cada tipo de formato de atributo que o Esquema de Dados suportar. Cada arquivo de atributo conterá então registros que especificam o

objeto ou relacionamento ao qual aquele atributo está associado, e o valor indicado para o atributo.

Como o atributo de identificação de um objeto sempre tem o mesmo formato em todos os objetos, esse atributo especificamente será armazenado em um campo do Arquivo de Objetos, e se for o único atributo com esse tipo de formato, um arquivo correspondente poderá ser omitido.

Em uma implementação tradicional de uma Base de Dados usando o Modelo Entidade-Relacionamento, considera-se a existência de entidades (entity) e de Arquivos de Entidades (entity set). No Modelo de Representação de Objetos passa a existir apenas um Arquivo de Objetos, e por isso o conceito de pertinência de uma entidade a um Arquivo de Entidades deve ser substituído pelo conceito de associação de um tipo a cada objeto. Dessa forma, a cada entidade de um dado Arquivo de Entidades no Modelo Entidade-Relacionamento corresponde sempre o mesmo Tipo de Objeto no Modelo de Representação de Objetos, e esse tipo é exclusivo dos objetos correspondentes às entidades de um mesmo arquivo, ou seja, o conceito de Arquivo de Entidades é substituído pelo conceito de Tipo de Objeto. O Tipo de Objeto de cada objeto é guardado no (único) Arquivo de Objetos da Base de Dados, e pode ser encarado como um atributo intrínseco de qualquer objeto. Nas Figuras 9 e 12 pode ser notado que o Tipo de cada Objeto é indicado em um campo de atributo apropriado do Arquivo de Objetos.

O mesmo ocorre com os relacionamentos, onde os vários Arquivos de Relacionamento (relationship set) são substituídos por apenas um, e surge o conceito de Tipo de Relacionamento, sendo que este será armazenado em um campo específico no (único)

Arquivo de Relacionamentos, como um atributo intrínseco de qualquer relacionamento, o que também pode ser visto por exemplo na Figura 12.

No Modelo de Representação de Objetos, os diversos atributos que podem ser associados aos objetos e aos relacionamentos também passam a ter significados diferentes daqueles que têm no Modelo Entidade-Relacionamento. O conceito de Atributo do ME-R, passa a ter o significado de Tipo de Atributo no MRD, e o Valor de Atributo pode ser chamado apenas de atributo. Dessa forma, um Tipo de Objeto passa a ter vários Tipos de Atributos, e os valores que eles assumem para cada objeto daquele tipo são os atributos daquele objeto.

O Modelo de Representação de Objetos proposto tem dessa maneira, uma estrutura de dados de acesso à informação composta basicamente de um Arquivo de Objetos, um Arquivo de Relacionamentos, e dos Arquivos de Atributos. Além disso modela a informação de um sistema classificando os objetos do sistema em alguns poucos Tipos de Objetos, os relacionamentos existentes entre os objetos em alguns poucos Tipos de Relacionamentos, e os atributos dos objetos e dos relacionamentos em alguns poucos Tipos de Atributos. A Figura 13 apresenta um quadro de como esses conceitos são denominados no âmbito dos modelos ME-R e MRD.

Denominação no ME-R	Denominação no MRD
Entidade	Objeto
Arquivo de Entidades	Tipo de Objeto
--	Código de Objeto
Atributo de Identificação	Nome de Objeto
--	Arquivo de Objetos
Relacionamento	Relacionamento
Arquivo de Relacionamento	Tipo de Relacionamento
--	Arquivo de Relacionamentos
Valor de Atributo	Atributo
Atributo	Tipo de Atributo
--	Valor de Atributo
Domínio de Atributo	--

Figura 13 - Correspondência das denominações de conceitos
no ME-R e no MRD.

III.4. - Maior Especificidade em Características de Tipos de Atributos

Na seção anterior, os tipos de atributos dos objetos de um Esquema de dados foram agrupados segundo padrões de formato de dados, que devem ser estabelecidos ao se criar um Esquema de Dados para uma aplicação de Meta-Sistema, ou seja, onde esse Esquema não pode ser totalmente definido a priori. No entanto, o agrupamento segundo diferentes formatos não acrescenta nada em relação a um diferente significado que algum tipo de atributo possa ter.

O significado adicional dado ao tipo de atributo de identificação dos objetos e o fato do formato desse tipo de atributo ser agora reconhecido como invariante em toda a Base de Dados traz, como foi notado, um maior conteúdo semântico a esse modelo. Com o intuito de aumentá-lo ainda mais, pode-se procurar identi-

ficar características comuns a outros grupos de tipos de atributos, de forma a possibilitar que um Sistema de Gerenciamento de Base de Dados trate mais especificamente os tipos de atributos de significado mais conhecido.

Os vários significados que os tipos de atributos possuem, refletem-se nos tipos de dados que são usados para representá-los, no mesmo sentido que o termo "tipo de dados" assume em linguagens de programação. Cada tipo de dado necessário à uma dada aplicação irá implicar em um arquivo na Base de Dados, como se viu. Mais do que isso, as operações associadas a cada tipo de dado irá obrigar que o Sistema de Gerenciamento de Base de Dados reconheça e suporte essas operações.

Não é importante, para efeito deste trabalho, quais são os Tipos de Dados e quais as operações que os suportam numa dada implementação de um Sistema de Gerenciamento de Base de Dados. O que é importante é reconhecer que alguns significados de determinados Tipos de Atributos possam ser reconhecidos e indicados para o SGBD. Por exemplo, em uma aplicação de PAC/FAC um atributo comumente encontrado é o que se refere à coordenadas de posicionamento. Atributos desse tipo podem ser representados através de um tipo de dado Vetor-Real-de-Três-Posições, porém apesar de ser descrita por três valores reais, coordenada de posicionamento traz muito mais significado do que o simples reconhecimento desses três valores apenas como números.

Para um Sistema de Representação de Dados de Engenharia, existem vários tipos de atributos cujo significado pode ter o reconhecimento do SGBD. A cada significado que um SGBD pode reconhecer, independentemente de um Tipo de Dado associado, será denominado Característica de Tipo de Atributo. Dessa forma, a

cada Tipo de atributo estará associado um Tipo de Dado e uma Característica, os dois constituindo o Significado associado ao Tipo de Atributo. Entre as Características de Tipo de Atributo mais comuns a um Sistema de Gerenciamento de Dados de Engenharia podem ser citadas as seguintes:

- Sinônimo. Corresponde a um identificador alternativo para um objeto;
- Comentário. Corresponde a informações para o usuário, que não são analisadas pelo sistema;
- Posição. Corresponde a uma informação de localização no espaço, podendo indicar um ponto ou uma orientação, ou ambos.
- Tempo. Indicação de tempo associado aos objetos e relacionamentos. Pode ser uma informação de tempo absoluto ou relativo. Podendo também apresentar várias formas, desde especificação de frações de unidades de tempo até indicações de data;
- Conjunto de Dados. São agrupamentos de informações semelhantes, que normalmente são representados em estruturas de arranjos de dados (vetores e matrizes), mas sobre os quais, podem ser definidas operações próprias de conjuntos, tais como união e intersecção, ordenação e tratamento de repetição de valores;
- Procedimento. São fórmulas ou parâmetros associados a atividades que devem ser exercidas de uma forma pré-determinada pelo sistema. Podem ser, por exemplo, atributos que tenham seu valor calculado a partir de valores de outros atributos do próprio objeto ou relacionamento a que está associado esse atributo, ou a

partir de valores "herdados" de outros objetos que se relacionam com o objeto a que esse atributo está associado;

- Regra. São atributos que possuem uma condição e uma ação associados. Sistemas de Gerenciamento de Dados que estejam acoplados a sistemas de gerenciamento de conhecimento serão beneficiados se puderem associar regras de conhecimento expressas na base de informações. Por exemplo, um objeto que seja do tipo atividade pode ser ativado quando uma regra indicar essa operação.

Tipos de Atributos que não tenham alguma característica especial que os distingua dos demais serão considerados tipos de atributos normais, referenciados simplesmente como **Propriedades** dos objetos ou dos relacionamentos a que estão associados.

Dos tipos de atributos que apresentam alguma característica especial, devem ter um tratamento especial para a implementação de um Sistema de Gerenciamento de Bases de Dados aqueles que se apresentam como atributos alternativos para a identificação de objetos, caracterizando sinônimos para o nome de um objeto, e por isso serão discutidos com mais detalhes. Os demais, para efeito de uma apresentação geral do conceito do reconhecimento de uma Característica de Atributo pelo SGBD, serão exemplificados aqui através de uma discussão da Característica Comentário, a qual corresponde à possibilidade de um atributo armazenar textos longos, não estruturados, para efeito de acesso pelo Sistema de Gerenciamento de Bases de Dados, caracterizando comentários sobre os objetos e/ou relacionamentos.

Atributos que tenham a característica de sinônimos do nome

de um objeto podem ser usados como uma forma alternativa de identificar um objeto, e são usados de forma independente do nome, que então passa a ter um sentido de nome principal, mas não tem mais o sentido de identificador exclusivo do objeto. Alguns exemplos de Tipos de Atributos com característica de Sinônimos podem ser: abreviatura do nome, os diversos códigos que podem ser atribuídos aos objetos, nomes alternativos ou em outro idioma etc.

Sendo distinguidos de outros Tipos de Atributos, os que têm característica de sinônimos podem ser separados dos demais e a própria estrutura de dados de acesso à informação pode ser alterada para refletir essa característica adicional. Em primeiro lugar, pode ser notado que mesmo não sendo o nome principal de um objeto, o seu formato deve ser o mesmo de um identificador. Isso permite que um Sinônimo ou o Nome principal de um objeto seja tratado sempre de maneira uniforme e comum. Em segundo lugar, o acesso lógico à informação armazenada sobre um determinado objeto pode ser feito identificando-se o objeto indiferentemente por seu Nome ou um seu Sinônimo, o que obriga a que ambos, Nome e Sinônimos sejam armazenados na mesma estrutura de dados, ou seja, os Sinônimos não podem ser armazenados em Arquivos de Atributos comuns, mas devem ser armazenados no mesmo arquivo onde estarão os Nomes de todos os objetos.

Tipos de atributos que tenham a característica de Comentários, sejam eles atributos de objetos ou de relacionamentos, terão como valor um texto, que poderá armazenar informações que não são estruturadas segundo o modelo de dados da base de dados. Exemplos de tipos de atributos com característica de Comentário são observações sobre um objeto ou relacionamento, anotações,

formatos de comandos sofisticados para o programa de aplicação, etc. Em uma base de dados de engenharia especialmente, esses atributos podem servir para explicações adicionais compreensíveis pelo usuário, ou para armazenar padrões de textos que podem ser completados ou modificados pelo programa de aplicação para gerar saídas do programa.

Tipos de Atributos com característica de comentário também devem ser tratados de forma diferente da que se usa para tratar os Tipos de Atributos que tenham outras características, pois estes requerem estruturas de dados capazes de armazenar textos longos, presumivelmente envolvendo muitas linhas de caracteres, e sendo reconhecidas subdivisões nos dados, tais como linhas e palavras. Além disso, Comentários necessitam de outras operações elementares próprias de editores de texto, que não são necessárias aos demais Tipos de Atributos que não tenham a característica de Comentários.

Nota-se assim que Sinônimos e Comentários são Características de Tipos de Atributos, que trazem a necessidade de operações elementares de acesso à informação, operações essas não necessárias para o restante das informações armazenadas na Base de Dados. Essas operações adicionais, se bem que aumentem a complexidade do modelo, refletem situações do mundo real que passam a ser representadas no modelo, aumentando correspondentemente o seu conteúdo semântico.

Da mesma maneira, as demais Características de Tipos de Atributos, tais como Posicionamento, Conjunto de dados, Procedimento ou Regra irão requerer a existência de arquivos de dados especiais, bem como a adição de operações elementares de acesso e manipulação das informações que levem tais

características. No entanto é difícil conceber a existência de uma implementação que possa incluir recursos de acesso e manipulação de todas as características citadas simultaneamente.

A escolha de quais Características de Tipos de Atributos serão contempladas por uma determinada implementação de um Sistema de Representação de Dados para Engenharia depende de quais características são necessárias, e quais são preferivelmente contempladas no SGBD e não no Programa de Aplicação, provavelmente usando as facilidades disponíveis em uma linguagem de programação de suporte. Novas técnicas têm surgido que procuram expandir a utilização de tipos de dados já existentes. Alguns trabalhos têm sido feitos desenvolvendo o conceito de "Tipos de Dados Abstratos" (ADT) com o objetivo de desenvolver critérios para a escolha e definição de um conjunto de estruturas de dados e procedimentos que manipulem todo o processamento necessário a determinadas estruturas de dados. Os conceitos desenvolvidos por estes trabalhos podem ser aplicados por um projetista de Sistemas de Gerenciamento de Bases de Dados para efetuar a escolha de quais Características de atributos um determinado SGBD em particular deverá suportar. Em especial, Gutttag [GUT_77] [GUT_78] tem apresentado técnicas que permitem agrupar informações estruturadas de diversas formas e apresentam regras de como podem ser escolhidas representações e implementações que sejam o mais abrangente possível [FOR_85].

Para efeito de facilitar a discussão, no restante deste trabalho será considerado que o sistema que se tem como objetivo pode se restringir a Tipos de Atributos que tenham características de Sinônimo, Comentário, Regra e Propriedades (atributos sem alguma característica específica). Pode-se considerar sumariamen-

te que as demais características citadas (Posicionamento, Tempo, Conjunto de Dados e Procedimento) podem ser consideradas como Propriedades ou Comentários pelo SGBD, e tratadas de modo especial pelo Sistema de Aplicação. O reconhecimento dessas características pelo SGBD pode ser feito seguindo o mesmo padrão desenvolvido para as características apresentadas.

III.5. - Escopo de Abrangência dos Identificadores de Objetos.

No Modelo Entidade-Relacionamento, o atributo (ou os atributos) usado para identificar uma entidade deve poder identificar uma entidade univocamente no arquivo daquela entidade. Entidades pertencentes a arquivos distintos podem ter um identificador comum sem que isso represente um problema. O mesmo vale para os identificadores de relacionamentos, sendo nesse caso não só um fato que pode ocorrer, mas sim que deve ocorrer, pois é dessa maneira que se estabelecem os relacionamentos entre as entidades individualmente.

Considerando, por exemplo, o caso de entidades, pode-se acrescentar que se um atributo não é suficiente para identificar univocamente uma determinada entidade, pode-se considerar a combinação de mais atributos, de forma a se chegar em um conjunto de atributos que constituam a chave de acesso primária de um Arquivo de Entidades.

No Modelo de Representação de Objetos, tal como proposto até este ponto, obrigatoriamente apenas um campo deve identificar cada objeto, e como todos os objetos são mantidos em uma única

estrutura de dados, conclui-se que cada objeto deve ter um Nome (e tantos Sinônimos quanto necessário) que seja único em toda a Base de Dados.

Claramente nota-se que deve ser acrescentado ao modelo a capacidade de restringir o escopo de abrangência de um identificador usado para localizar um objeto.

Em um sistema de representação de dados para engenharia, é desejável que o modelo de dados esteja bastante próximo do que o usuário "sente" que seja a informação que a Base de Dados manipula. A forma de um modelo concatenar atributos para obter a chave de acesso do ME-R não é tão "natural" quanto um modelo que permita aumentar ou diminuir a especificidade de um identificador, como se a "visão" que se tem da informação pudesse ser mais ou menos detalhada, a critério do próprio usuário. Dessa forma, é preferível dotar o MRO de capacidade para refletir esse último tipo de comportamento.

Pode-se dizer que um sistema, quando visto de maneira global, apresenta um pequeno grau de detalhe. Quando o interesse recai em uma de suas componentes, o grau de detalhe aumenta, porém toda a informação que então se apresenta refere-se diretamente àquela componente que está sendo observada. Pode-se então pensar em uma ferramenta de comunicação com o usuário, que interprete qualquer identificador dessas informações, levando em consideração somente o conjunto de informações que está sendo examinado, e exigindo que qualquer referência a outras informações tenha um teor de identificação maior.

Pode-se dotar o modelo proposto dessa capacidade se for considerado que, quando se manipula a base de dados de um ponto de vista global, são poucos os objetos acessíveis. Indicando-se

ao sistema que se pretende considerar a seguir um desses objetos, indica-se uma região da Base de Dados em que se pretende obter ou manipular informações. Dessa forma tornam-se acessíveis novos objetos. Esse procedimento pode ser repetido, indicando-se para consideração um dos novos objetos tornados disponíveis, dessa forma aumentando-se sucessivamente o nível de detalhe, e correspondentemente a quantidade de objetos a que se tem acesso diretamente através de seus Nomes.

Cada vez que se coloca em consideração um objeto, aumenta-se um nível de detalhe, porém não são todos os objetos desse nível que se tornam disponíveis, mas apenas aqueles que detalham mais o objeto considerado. Por exemplo, se o Sistema de Gerenciamento de Bases de Dados está sendo usado para se armazenar os dados de um projeto de engenharia, em um primeiro nível de detalhe seriam reconhecíveis as várias atividades, metas e recursos desse projeto. Quando passa-se a considerar uma dada atividade, os objetos que especificam essa atividade tornam-se acessíveis, porém não os que especificam as demais atividades.

O modelo proposto pode então contar com essa capacidade se lhe forem incorporados os conceitos de Classe de Objetos, Objeto Considerado, e Conjunto de Objetos Constritos por um Objeto Considerado.

Os objetos de uma Base de Dados passam a ser acessíveis em blocos, sendo que cada bloco se refere não somente a um nível específico de detalhe de informação, mas também a um determinado aspecto de informação. Por exemplo, na Base de Dados de um projeto de engenharia, quando leva-se em consideração um maior detalhe de uma atividade, não se aumentam correspondentemente os detalhes dos recursos usados por essa atividade, mas tão somente os deta-

lhes da própria atividade. Se desejado, o usuário poderia aumentar o nível de detalhe de um recurso de sua escolha, identificando para consideração esse recurso, e dessa forma teria então o detalhe de uma atividade e o de um recurso simultaneamente, mas sempre apenas um de cada num mesmo instante.

Para isso, os objetos de uma Base de Dados devem ser classificados segundo diferentes aspectos, sendo essa classificação feita usando os diferentes Tipos de Objetos do Esquema de dados. Dessa forma, cada objeto pertence a apenas uma Classe de Objetos, determinada através do seu tipo de objeto.

O conceito de Classes de Objetos divide os objetos de uma Base de Dados em vários aspectos de interesse, que são definidos pelos Tipos de Objetos que compõem o Esquema de Dados daquela base. Pode-se considerar então que o conceito de Classes de Objetos estabelece uma partição no conjunto de Tipos de Objetos. Uma das classes terá que estar permanentemente disponível, e será denominada Classe Global. Os objetos da Classe Global são os que estão disponíveis quando se está no nível menos detalhado de acesso ao sistema, e permanecem sempre disponíveis.

Objeto Considerado é um objeto que se tem presentemente disponível, e que se indica para consideração, o que faz com que os objetos que o especificam com maior grau de detalhe se tornem também disponíveis.

Quando considera-se um objeto, indica-se implicitamente uma classe de objetos. Como foi visto, não são todos os objetos da classe indicada que se tornam disponíveis, mas apenas os que pertencem ao Conjunto de Objetos Constritos pelo objeto indicado para consideração.

Indicar um objeto para consideração caracteriza um operação

de Abrir uma Classe de Objetos para acesso, tornando acessíveis os objetos do Conjunto de Objetos Constritos pelo Objeto Considerado. Essa abertura pode permitir que os objetos se tornem acessíveis para consulta e modificação ou apenas para consulta.

Quando se indica para consideração um objeto que indica uma Classe de Objetos já aberta, esta é fechada, o que faz com que o Conjunto de Objetos Constritos anteriormente deixe de estar acessível, e a seguir a mesma Classe é novamente aberta, agora tornando disponíveis os objetos constritos pelo novo Objeto Considerado. Outros objetos, pertencentes a classes que não aquela focalizada implicitamente pela indicação de um novo objeto para consideração, e a correspondente operação de abertura de classe, não são afetadas por essa operação.

Os Objetos da Classe Global estarão sempre disponíveis, não sendo possível fechar essa classe.

Se o usuário desejar referir-se a qualquer dos objetos que estão disponíveis em qualquer uma das classes abertas, o Nome do objeto e talvez a indicação de seu tipo será suficiente para identificá-lo. No entanto, se for necessário se referir a algum objeto pertencente a uma classe não aberta, ou a um objeto pertencente a um conjunto de objetos constritos por um objeto que não o presentemente considerado, então será necessário que se identifique todo o caminho de acesso, desde o ponto de interesse atual na base de dados até o ponto onde está o objeto a que se quer referir.

Com esses conceitos, o Modelo de Representação de Objetos passa a permitir que objetos que pertencem a Conjuntos Constritos por objetos distintos tenham nomes iguais, e mesmo assim possa se manter a sua identificação inequívoca, através da identificação

Figura 14 - Esquema de Dados de um Laboratório de Projetos de Sistemas Digitais

simultânea do ponto de interesse do usuário em um dado instante.

Os conceitos apresentados podem ser exemplificados usando-se uma Base de Dados para armazenar dados para o laboratório de projetos de sistemas digitais. Na Figura 14 é apresentado um Esquema Conceitual de uma base de dados para esse fim, um pouco mais abrangente do que o iniciado na Figura 10.

Pode-se perceber que os RECURSOS, os PROJETOS e as PESSOAS de um laboratório são sempre reconhecidos no âmbito de um laboratório, e por isso podem ser colocados na Classe Global. Já as METAS e ATIVIDADES dizem respeito apenas a cada respectivo PROJETO, e portanto são objetos que devem ser colocados em uma classe própria, que poderia ser denominada Projeto. Dentro de uma atividade são produzidas várias PLACAS-DE-CI, com seus respectivos CIs, as quais dizem respeito apenas a cada ATIVIDADE, e por isso constituem a classe de Atividades. A tabela da Figura 15 mostra como os vários tipos de objetos podem ser classificados.

Classe	Tipos de Objetos
GLOBAL	RECURSO, PESSOA, PROJETO
Recurso	INSTRUMENTAL, DOCUMENTAÇÃO
Projeto	ATIVIDADE, META
Atividades	PLACA-DE-CI, CIRCUITO-INTEGRADO TIPO-DE-CIRCUITO-INTEGRADO
Metas	PRODUTO-FINAL

Figura 15 - Classe de cada Tipo de Objeto

Usando esse Esquema de Dados, o usuário terá sempre disponível os PROJETOS, PESSOAS e RECURSOS do laboratório. Quando

quiser, poderá acessar as informações de um determinado projeto, digamos o projeto com o código P86/2, indicando para consideração esse projeto, o que torna todas as ATIVIDADES e METAS desse projeto acessíveis. ATIVIDADES de outro PROJETO que não o P86/2 não estarão diretamente acessíveis, e somente poderão ser acessadas se o seu Nome ou Código forem indicados explicitamente junto com o Nome do PROJETO a que pertencem. No entanto, o usuário poderá obter simultaneamente detalhes de algum RECURSO, digamos sobre a Linha-de-Banhos-Para-Placas, indicando-a para consideração.

Nesse instante, todos os objetos dos tipos PROJETO, PESSOA, e RECURSO da Base de Dados estarão disponíveis, bem como todos os objetos dos tipos INSTRUMENTAL e DOCUMENTAÇÃO constrictos pelo objeto P86/2, e os objetos dos tipos ATIVIDADE e META constrictos pelo objeto Linha-de-Banhos-Para-Placas.

Se o usuário agora se referir a uma ATIVIDADE Desenvolver-Terminal-Gráfico, e indicá-la para consideração, ele passará a ter disponíveis também os objetos do tipo PLACA-DE-CI e CIRCUITO-INTEGRADO constrictos por aquela ATIVIDADE. Se nesse instante o usuário se referir a uma PLACA-DE-CI chamada Processador-de-Sinal-Analógico, o sistema entenderá que se refere a uma PLACA-DE-CI da ATIVIDADE Desenvolver-Terminal-de-Video do PROJETO P86/2.

Se agora o usuário indicar para consideração um outro PROJETO, digamos o P83/5, os objetos do tipo ATIVIDADE e META que estavam disponíveis deixarão de estar, e passarão a estar disponíveis aqueles que são constrictos pelo PROJETO P83/5. Além disso, todos os objetos dos tipos PLACA-DE-CI e CIRCUITO-INTEGRADO deixarão de estar disponíveis, pois o objeto que os constringia

também já não estará disponível. No entanto, os objetos dos tipos INSTRUMENTAL e DOCUMENTAÇÃO não serão afetados, uma vez que o objeto que os constrói (Linha-de-Banhos-Para-Placa) continua disponível.

III.6. - Identificação de Objetos Através de Relacionamentos

A adoção do conceito de Classes de Objetos exposto na seção anterior permite que se defina um escopo de abrangência para os nomes e sinônimos de objetos, permitindo dessa forma o acesso seletivo de objetos através desses nomes e sinônimos. No entanto não é apenas através dos identificadores de objetos que se pode acessá-los, pois pode-se obter acesso a um dado objeto através dos relacionamentos que o envolvam. Assim, partindo-se de um objeto qualquer, possivelmente identificado por seu nome ou um sinônimo, pode-se obter acesso a um objeto que esteja a ele relacionado, então a partir deste acessar outro a ele relacionado, e assim por diante, navegar toda a base de dados, sem que necessariamente se acessem outros objetos através de um identificador.

No modelo tal como proposto até aqui, todos os objetos deveriam necessariamente ter um nome. No entanto, em um Sistema de Gerenciamento de Dados para Engenharia, existem situações onde um objeto pode não ter um nome significativo, que poderia então ser omitido da Base de Dados. Num caso assim, o objeto deveria ser acessado apenas através de relacionamentos que o envolvam.

Essa necessidade pode ser solucionada no modelo proposto se for incluído nele o conceito de Objeto Identificado por Relacionamento, que caracteriza exatamente esse caso.

Na Figura 15 ocorre essa situação no caso de uma PLACA-DE-CI que USA determinados TIPOs-DE-CIRCUITO-INTEGRADOs, sendo mesmo que se pode usar vários CIRCUITO-INTEGRADOs de um mesmo tipo. Mas apesar de cada TIPO-DE-CIRCUITO-INTEGRADO ter um nome (por exemplo SN7400), e até possuir uma documentação, cada CIRCUITO-INTEGRADO em si, ou seja a sua ocorrência na PLACA-DE-CI não tem necessariamente um nome (pode-se designá-lo por exemplo como U1, U2, etc, mas isso nem sempre é obrigatório). Usando o conceito de identificar o CIRCUITO-INTEGRADO por um relacionamento PLACA-DE-CI USA TIPO-DE-CIRCUITO-INTEGRADO, além de se poder prescindir de um nome para uma ocorrência, faz-se uma ligação semântica bastante forte entre o CIRCUITO-INTEGRADO e o seu TIPO-DE-CIRCUITO-INTEGRADO. Na Figura 15 essa ligação está simbolizada pela linha de asteriscos entre o Tipo de Relacionamento USA e o Tipo de Objeto CIRCUITO-INTEGRADO.

O uso de uma ligação entre um Objeto e um relacionamento, com o objetivo de se fazer a identificação do objeto através do relacionamento facilita a correlação de dados dos objetos envolvidos pelo relacionamento (Objetos Ascendentes) com os dados do objeto identificado pelo relacionamento (Objeto Descendente). Isso é uma ocorrência comum, principalmente no uso de um Sistema de Gerenciamento de Dados para Engenharia. Um objeto ser identificado por um relacionamento corresponde ao fato de o Objeto Descendente herdar dados dos Objetos Ascendentes.

Por exemplo, um determinado CIRCUITO-INTEGRADO contém todos os parâmetros explicitados pelo TIPO-DE-CIRCUITO-INTEGRADO que

ele é. No entanto, os dados (Atributos) de um TIPO-DE-CIRCUITO-INTEGRADO não devem ser transcritos para cada ocorrência de um CIRCUITO-INTEGRADO, mas essa informação deve apenas ser herdada. Os Atributos próprios de cada ocorrência de um CIRCUITO-INTEGRADO são os únicos que devem ser associados a ele, como por exemplo sua POSIÇÃO na PLACA-DE-CI.

III.7. - O Uso do Modelo de Representação de Objetos em um Ambiente Multiusuário.

Um Sistema de Gerenciamento de Base de Dados em geral é concebido para atender a um grupo de usuários, com perspectivas e necessidades diferentes sobre os dados armazenados na Base de Dados. As diferentes necessidades de cada usuário, dentro das possibilidades do SGBD, normalmente são supridas através de um esquema de proteções de acesso, permitindo a cada usuário o acesso somente às informações que lhe dizem respeito, bem como permitindo o uso apenas das operações do SGBD que são necessárias. Um SGBD que se apóie no Modelo de Representação de Objetos (MRO) também irá distribuir o acesso às informações e às operações através de um esquema de proteções semelhantes.

Quanto às diversas perspectivas dos usuários, representadas nas várias visões de dados que compõem o esquema de dados, elas são contempladas por um SGBD que adote o Modelo de Representação de Objetos de uma forma diferente dos demais, por dois motivos.

Em primeiro lugar, em um modelo tradicional, tal como no modelo de Rede, no relacional ou no próprio ME-R, os vários

campos (ou atributos) são vistos pelo usuário compondo registros, que nem sempre estão em um mesmo registro no esquema geral da Base de Dados. Isto é, um conjunto de informações na visão de um usuário pode compor um único registro, que tanto pode ser um subconjunto das informações de um único registro no esquema geral, como pode conter informações que no esquema geral estão em mais de um registro.

No MRD não existe o conceito de um registro de informações, pois cada objeto é tratado de forma individual. O que existe em seu lugar, é o conjunto dos valores assumidos pelos atributos e os relacionamentos de cada objeto. Dessa forma, no esquema geral da Base de Dados, para cada Tipo de Objeto estarão associados todos os Tipos de Atributos e todos os Tipos de Relacionamentos com todos os respectivos Tipos de Objetos Destino. Cada visão do usuário terá acesso a um subconjunto de Tipos de Objetos do conjunto de Tipos de Objetos do esquema geral, e além disso, para cada Tipo de Objeto cujo acesso é permitido, cada visão de usuário terá acesso a um subconjunto dos Tipos de Atributos e de Relacionamentos que aquele Tipo de Objeto tem no esquema geral.

Além disso, como o MRD inclui o conceito de Classes de Objetos, pode-se refinar ainda mais o procedimento de se permitir o acesso mesmo entre os objetos de um só Tipo de Objeto. O conceito de Classe de Objeto do MRD estabelece uma partição no conjunto de objetos de um determinado tipo. Esse efeito corresponde a uma partição estabelecida entre as linhas (registros) dos arquivos das bases de dados tradicionais, porém no MRD essa partição não é feita através dos valores que os campos de um arquivo podem assumir, como é o caso dos modelos tradicionais, mas sim por uma estrutura específica, interna à estrutura de

dados física e ao caminho de acesso interno da base de dados.

Conforme foi visto, a implementação de um SGBD que adote o travamento de dados a nível de registros físicos é bastante custosa: tanto em termos do programa de aplicação, que deve informar ao SGBD sempre que, após ter solicitado uma consulta à uma informação, deixar de usá-la; quanto em termos do SGBD, que deve manter um registro de todos os dados passados para algum programa de aplicação e para os quais ainda não tenha havido sinalização de que tais dados não mais estão sendo usados.

A implementação de um SGBD que use o MRD pode sem dúvida adotar a proteção a nível de registros físicos. No entanto o modelo fornece meios para evitar que isso seja necessário. O conceito de Classe de Objeto pode ser usado, de forma que quando uma classe é aberta, pode-se abri-la apenas para consulta ou para atualização, de forma que a proteção pode ser feita a nível de classes. Essa é uma forma de proteção baseada em travamento de dados por predicado, porém no MRD é muito mais simples de ser realizada do que nos sistemas tradicionais, ao mesmo tempo em que é muito mais natural, pois o SGBD deve manter registro apenas das classes que estão abertas para cada usuário, e se estão abertas para atualização ou apenas para consulta. O programa de aplicação não precisa informar nada mais além do fato de que uma classe deve ser fechada, o que pela própria lógica da aplicação, já deve ocorrer naturalmente.

Esse esquema de proteção, além de ser mais simples, tem uma correspondência direta com a operação de alto nível requisitada pelo programa de aplicação, uma vez que se reflete nas operações dos usuários do programa de aplicação. Por exemplo, se um usuário estiver usando um editor para modificar o conteúdo da base de

dados, ele provavelmente estará trabalhando com uma parte da base total, restrita a alguma classe de objetos local ao seu ponto de interesse num dado instante. Com a proteção atuando a nível de classes, a classe onde ele estiver trabalhando estará protegida do acesso por qualquer outro usuário, ao mesmo tempo em que não poderá ter acesso a uma área de interesse sendo acessada por outro usuário.

Um ponto de conflito para esse tipo de proteção está na classe global, à qual todos os usuários deverão obrigatoriamente ter acesso, para que qualquer outra classe possa ser acessada. No entanto, se for considerado que em uma base correspondente a um projeto já adiantado, envolvendo vários usuários, a classe global já deverá estar definida e razoavelmente estática, então qualquer usuário poderá ter acesso a ela apenas para consulta, permitindo que muitos usuários compartilhem a permissão de consulta à classe global, e apenas em classes em nível hierárquico mais baixo cada usuário obterá permissão individual para atualizações. A atualização da classe global passa assim a ser permitida apenas a usuários privilegiados, tal como o gerente da aplicação que usa aquela Base de Dados, o qual para atualizá-la deverá requerer que naquele instante ninguém mais tenha acesso à qualquer porção da Base de Dados (o que aliás é natural).

III.7.1. - O Uso do MRD em um Ambiente de Multiprocessamento.

O termo "Multiprocessamento" é com freqüência empregado para designar dois tipos de arquitetura de sistemas de computadores, que envolvam mais de uma unidade de processamento. Um dos tipos

corresponde às arquiteturas para processamento distribuído, onde as unidades de processamento constituem sistemas de processamento independentes, com seus recursos próprios de memória principal e secundária, com uma taxa de transferência de informações entre unidades de processamento relativamente baixa. A outra arquitetura corresponde a um único sistema de processamento, com as várias unidades de processamento partilhando ao menos parte da memória disponível, e com uma taxa de transferência de informações entre unidades muito elevada.

O uso de uma base de dados em um sistema de processamento distribuído tem sido amplamente estudado, e as suas variadas características e consequências estudadas e discutidas na literatura. Pode-se citar entre os trabalhos recentes [SAR_85], onde se discutem problemas que existem quanto à disponibilidade de acesso a dados distribuídos em uma rede, e se apresenta a sugestão de uma arquitetura de programas distribuídos que procura minimizar os problemas, e [SHE_85], onde é apresentado um modelo de representação de transações de dados que ocorrem em Sistemas de Base de Dados Distribuídas. Quanto a Bases de Dados envolvendo modelagens de dados de alto conteúdo semântico, [WSU_86] e [KEL_86] apresentam algumas considerações e sugestões interessantes.

O Modelo de Representação de Objetos, de uma maneira geral, tem o mesmo comportamento que os demais modelos, quando usado em um Sistema de Base de Dados Distribuído, e por isso esse aspecto não será discutido aqui.

O tipo de multiprocessamento para o qual este modelo apresenta características peculiares é o caracterizado pela possibilidade de um processamento paralelo de informações, com uma alta disponibilidade das informações, isto é, não importa se o para-

lelismo é conseguido através de um sistema distribuído ou não, mas sim que o custo de transferência de informações seja despre-sível. Isso caracteriza o segundo tipo de multiprocessamento.

O Modelo de Representação de Objetos, modelando cada objeto individualmente, oferece a possibilidade de se permitir que um SGBD, que use tal modelo, aceite de um usuário instruções para executar alguma atividade quando um objeto é acessado de alguma forma pré-estabelecida. Ou seja, é possível que se anote dinamicamente, em cada objeto ou grupamento de objetos, condições em que alguma atividade deva ser tomada. Isso pode ser feito tanto implicitamente pela aplicação, quanto explicitamente pelo usuário, através de atributos com características de Regras.

Por exemplo, pode-se solicitar que o gerente da configuração do projeto seja informado de qualquer tentativa de acesso a um determinado objeto que faça parte de uma Configuração Básica congelada, pode-se solicitar que, ao se fechar uma classe de objetos que tenha sido modificada, determinados testes sejam efetuados, ou pode-se solicitar que se crie uma seleção de objetos contendo todos os objetos que foram referenciados no dia de hoje, etc. Essas operações são notoriamente paralelas, no sentido de que a operação diretamente solicitada prossegue normalmente, sem que o usuário que a solicitou tome consciência de que essa operação disparou uma outra, cujas consequências não têm nenhum relacionamento com a operação solicitada pelo usuário, mas sim com a atividade previamente anotada.

Essa possibilidade oferecida pelo MRO será denominada Anotação para Acesso, formalmente especificada como constituindo-se de:

- um Conjunto de objetos sobre os quais a anotação é

estabelecida, que constitui o Conjunto de Abrangência da Anotação para Acesso;

- uma Condição, que é confrontada com a operação solicitada sobre o objeto;
- uma Atividade, que é executada sempre que a condição é satisfeita.

Em um Sistema de Representação de Dados de Engenharia, a propriedade de Anotação para Acesso é muito útil, devido principalmente ao grande volume de operações que normalmente se realizam, e à possibilidade que esse recurso abre de postergar operações que podem ser agrupadas para processamento posterior, permitindo ao programa tratar cada atividade separadamente.

Por exemplo, qualquer alteração efetuada na descrição de um sistema terá que ser submetida a um conjunto de testes, que irão avaliar, por exemplo, o impacto dessa alteração em partes não modificadas do sistema descrito (ver seção 4.5). Esses testes, para serem realizados pelo mesmo aplicativo que efetuou a mudança, no caso provavelmente um editor que o usuário estará operando, irá consumir tempo, além de exigir um aplicativo complexo o suficiente para poder realizar os testes.

A Anotação para Acesso efetuada nesse exemplo será constituída por:

- Conjunto de abrangência: qualquer objeto cuja descrição já houvesse sido aprovada.
- Condição: quando acessado para atualização.
- Atividade: deverá ser novamente submetido ao processo de teste para aprovação da descrição.

Usando o recurso de Anotação para Acesso, o editor pode simplesmente ignorar quaisquer testes que devam ser efetuados. Se houvesse sido anotado que qualquer objeto cuja descrição já tivesse sido aprovada quando acessado para atualização, deveria ser novamente submetido ao processo de teste para aprovação da descrição.

Em [TRA_82] é apresentado um conjunto de atividades que um sistema de representação de dados, que se usasse o Modelo de Representação de Objetos poderia executar em processos paralelos. As atividades sugeridas indicam que a Anotação para Acesso pode ser solicitada diretamente pelo usuário, ou automaticamente, por uma operação interna do sistema.

Aqui vale uma consideração sobre os Tipos de Atributos com Características de Regra. Uma regra é um atributo associado a um determinado objeto, sendo constituída de uma Condição e uma Atividade que é executada quando a condição é satisfeita. Pode-se notar que estão presentes as três partes que constituem uma Anotação para Acesso, com as seguintes diferenças:

- uma regra se aplica a apenas um objeto, e uma Anotação para Acesso pode se aplicar a mais de um;
- em uma regra, a condição diz respeito aos valores assumidos por variáveis do sistema e/ou pelos atributos do objeto a que se aplica, aos seus relacionamentos, etc., ao passo que numa Anotação para Acesso, a condição é sempre uma forma de acesso que tenha sido feita aos objetos a que se aplica;
- as atividades tanto das regras quanto das Anotações para Acesso podem ser vistas como pertencendo a um conjunto comum de atividades reconhecidas pelo par "Sistema de

A semelhança que existe entre a estrutura dos atributos com Característica de Regra, e a estrutura das Anotações para Acesso, sugere que o Sistema de Representação de Dados use uma estrutura única para dar suporte aos dois conceitos, desde que a estrutura seja flexível o suficiente para englobar as necessidades dos dois conceitos. Dessa forma, os atributos com característica de regras terão o seu valor (condição e atividade) armazenados na estrutura de dados da base de dados, e são dessa forma tratados como atributos. Uma vez que um aplicativo acesse um tal atributo e indique sua ativação, ele passa para a estrutura dinâmica de Anotação para Acesso, onde é levado em consideração sempre que um novo acesso for efetuado na base de dados. Assim pode-se dizer que a ativação de uma regra origina uma Anotação Para Acesso, e uma condição de uma Anotação para Acesso que seja satisfeita origina a ativação de uma Atividade.

Quanto à Abrangência de uma Anotação, é necessário que a estrutura de dados suporte a indicação de um conjunto de objetos sobre os quais se estabelece uma anotação de acesso. Quanto às condições, a estrutura de dados deve aceitar a indicação das condições tipicamente associadas às Anotações para Acesso tão bem quanto das associadas às regras. Por fim, a estrutura deve poder armazenar a indicação de qualquer atividade que o sistema possa executar.

A possibilidade de execução de processos em paralelo não obriga a que um tal modelo somente possa existir em uma arquitetura de processamento que permita o paralelismo de execução, pois essas atividades podem perfeitamente ser realizadas em seqüência,

em um sistema com apenas um processador. No entanto, nesse caso, é preferível que o sistema de representação de dados possa armazenar, para posterior processamento, as atividades ativadas através de condições satisfeitas de Anotações para Acesso, na forma de transações contendo todas as atividades necessárias. Isso é desejável por dois motivos: primeiro porque dessa forma a aplicação que está em andamento não tem seu tempo de execução atrasado, o que é muito importante se for uma aplicação interativa; segundo porque dessa forma, quando a aplicação terminar, as transações pendentes podem ser ordenadas, e executadas na melhor ordem possível.

O segundo motivo traz a necessidade de que sejam examinadas possíveis consequências da alteração da ordem com que as transações são executadas. Para que essa alteração seja válida, é necessário que todas elas sejam transações atômicas na base de dados, ou seja, é necessário que os efeitos obtidos pela execução de muitas operações concorrentemente sejam os mesmos dos obtidos pela execução das mesmas operações em alguma ordem seqüencial, e além disso, que nenhum efeito parcial seja visível, causado por alguma impossibilidade de execução de uma ação individual.

O resultado obtido processando um conjunto de transações a posteriori pode não ser o mesmo de uma execução em paralelo, porém mesmo sem que isso ocorra, o objetivo pelo qual se ativam transações em paralelo podem justificar o seu emprego. Consideremos por exemplo que a aplicação solicitada explicitamente pelo usuário corresponde a uma edição interativa. Nesse caso, as ações ativadas por acessos solicitados pelo editor correspondem às atividades de verificação da descrição resultante. A execução imediata de testes sobre a descrição, à medida que os acessos do

editor vão ocorrendo, irão frequentemente levar à constatação de que a descrição não está correta, pois o usuário não terá ainda completado todo o processo de edição idealizado. A execução posterior dos testes, sobre a descrição agora considerada encerrada pelo usuário irá detectar apenas as falhas não percebidas pelo usuário, que podem ser (e em geral o são) bem diferentes das que seriam detectadas pelo processo de testes executado de maneira concorrente. Isso mostra que o resultado obtido por se executar essas atividades como transações a posteriori é preferível, e permite caracterizar todo um processo de edição como apenas uma transação.

A conclusão do exemplo acima, sobre um sistema onde a execução de atividades automáticas é realizada após o encerramento de uma atividade solicitada pelo usuário, não implica que um sistema que admita concorrência não traz benefício para um sistema que emprega o MRD. Apenas indica que nem sempre as atividades que podem ser executadas em paralelo o devam ser. Por exemplo, quando o editor do exemplo acima encerrar a execução, todos os testes poderão ser executados em qualquer ordem, ou mesmo serialmente. Nesse ponto, a propriedade de ter um resultado sempre bem determinado existe entre as transações pendentes, e o processo todo poderá ser realizado mais rapidamente se executado em paralelo. Além disso, todo o processo de registro dos testes que devem ser postergados é feito para cada acesso que modifique a Base de Dados, e isso pode ser feito em paralelo com a própria modificação da Base de dados.

Disso conclui-se, que a implementação de um sistema usando o Modelo de Representação de Objetos, mesmo que seja feita em um sistema que admita processamento em paralelo, deverá contar com

uma estrutura que permita o acúmulo de indicações de transações para posterior execução. Isso porque devem ser postergadas para execução posterior ao encerramento de uma aplicação, todas as transações que forem indicadas para ativação através de Anotações para Acesso, cuja condição tornou-se satisfeita a partir de um acesso originado de uma transação que pode alterar a base de dados. A transação cujo término deve ser aguardado é justamente aquela que pode alterar o conteúdo da base de dados.

III.8. - Conclusões

Neste capítulo foi feita a apresentação do Modelo de Representação de Objetos partindo de conceitos oriundos do Modelo Entidade-Relacionamento (ME-R). Como foi esclarecido na apresentação do capítulo, o ME-R foi usado como referência para a apresentação que foi feita, porém o MRO constitui-se em um modelo autônomo, em todas as características que apresenta. Como uma verificação do modelo quanto à sua adequação às aplicações alvo é feita no capítulo seguinte, resta então situar o modelo entre os modelos tradicionais e semânticos discutidos no capítulo II.

O MRO é um modelo que pode ser caracterizado como um modelo semântico. Sob alguns aspectos, o ME-R também pode ser considerado assim, e alguns autores não hesitam em citá-lo como o primeiro modelo semântico desenvolvido. No entanto, com exceção do Modelo Entidade-Categoria-Relacionamento, todos os demais modelos semânticos apresentados são conceitualmente concebidos

tendo por base o Modelo Relacional (assim como o próprio ME-R).

A característica marcante que distingue o Modelo Relacional do ME-R é o fato de que o ME-R especifica as associações entre as relações de itens explicitamente, enquanto o Modelo Relacional indica as associações implicitamente, através da comparação de valores de atributos comuns a duas relações. Quase todos os demais modelos semânticos desenvolvidos usaram a associação implícita usada pelo modelo relacional. Não cabem aqui considerações sobre o motivo dessa omissão (maior complexidade para a modelagem algébrica do modelo resultante, inexistência de implementações já operacionais, etc.). O fato é que a representação explícita de relacionamentos traz para um modelo uma capacidade de representação semântica muito grande.

O trabalho de Smith & Smith [SMI_77b], foi que marcou a introdução do conceito de abstrações de generalização e de agregação, que passou a ser a base principal das formulações dos modelos semânticos. Esse trabalho, contemporâneo da apresentação do ME-R é baseado no modelo relacional, assim como o são todos os modelos que, a partir daquele trabalho, começaram então a ser propostos.

O MRO é também um modelo semântico, e admite também a representação de abstrações, tanto de generalização quanto de agregação, porém, como fica claro pelo desenvolvimento da sua apresentação neste capítulo, incorpora também a especificação de associações entre objetos explicitamente, através de relacionamentos, da mesma maneira que o ME-R. Dessa forma, as classificações das formas como associações entre relações são feitas nos demais modelos semânticos, passam no MRO a corresponder a classificações dos tipos de relacionamentos que são usados em uma modelagem. Por

exemplo, uma associação de agregação, que nos demais modelos deveria ser sempre representada através de uma estrutura sintática e semântica rígidas, apresenta no MRO a conveniência de poder ser representada como um relacionamento, com seu nome e estrutura estabelecidas de maneira flexível, pelo projetista da aplicação, que apenas indicaria o fato de um relacionamento ter características de agregação.

A explicitação de um relacionamento traz assim um aumento na capacidade semântica do modelo, cujo valor pode ser melhor compreendido se for analisado o fato de que com a sua introdução no modelo, os atributos, uniformemente tratados em todos os demais modelos (inclusive no ME-R), no MRO passam a ser divididos em pelo menos três categorias funcionais.

Primeiro, existe a separação dos atributos intrínsecos, que são mantidos pelo próprio SGBD. Dessa forma, informações que são obrigatoriamente mantidas a nível de aplicação nos demais modelos, no MRO são intrinsecamente manipuladas pelo próprio SGBD, tais como objeto origem e objeto destino. Atributos de relacionamentos são apenas aqueles que podem variar de tipo de relacionamento para tipo de relacionamento.

Segundo, dentre os atributos extrínsecos existe a separação das informações que são locais a cada objeto e relacionamento (atributos propriamente ditos), de todas as informações que caso contrário seriam necessárias apenas para manter as informações que permitissem a manutenção implícita de relacionamentos entre os objetos.

Neste capítulo a apresentação do MRO foi feita através das principais características que o distinguem dos demais modelos,

das características que são basicamente inovadoras em relação aos demais modelos existentes. As características que ocorrem normalmente nos demais modelos (e também no MRO), tal como os conceitos de abstração de agregação e de generalização, não foram ressaltadas neste capítulo, considerando-se normal o seu emprego. No capítulo seguinte todos os conceitos aqui apresentados, bem como os demais normalmente empregados em modelos semânticos são tomados conjuntamente, e o Modelo de Representação de Objetos tem o conjunto de todas as suas estruturas e operações uniformemente apresentadas.

O MRO apresenta vários conceitos novos em modelos semânticos, porém dois deles devem ser aqui ressaltados. O primeiro é a especificação de Características de Atributos, principalmente com as características de propriedade, sinônimo, comentário, regra, e gráficos, pois cada um deles traz implicações quanto à forma particular com que são individualmente armazenados e manipulados, e portanto devem ter suas particularidades reconhecidas e atendidas por um SGBD. Esse reconhecimento no entanto até hoje não foi feito em nenhuma implementação (e mesmo em nenhum modelo) de que tenhamos conhecimento.

O segundo conceito inovador em um modelo semântico é o conceito de Classes de Objetos. É comum que, em qualquer aplicação em que o significado dos dados seja importante, os dados sobre os quais são feitas referências sejam implicitamente localizados, implicação que é feita pela própria indicação do contexto onde se está trabalhando. O acompanhamento do foco de interesse de uma aplicação implicitamente pelo SGBD, inerente ao próprio conceito de classes de objetos, também não existe até hoje feito

de maneira natural em nenhum SGBD. A concatenação de chaves de identificação em modelos orientados para registros resolve o problema da identificação de registros, mas não reflete de maneira nenhuma o acompanhamento do foco de interesse (em alto nível semântico) que as classes podem fazer.

Fundamentalmente, os conceitos de Classes de Objetos, Características de Atributos, Relacionamentos entre Objetos explícitos, e abstrações de generalização, são conceitos que permeiam basicamente todo o MRD, e se refletem em todas as estruturas e operações que o constituem, e são também a fonte de toda a sua capacidade de modelagem.

CAPÍTULO IV

ESTRUTURA DE DADOS E OPERAÇÕES NO MODELO DE

REPRESENTAÇÃO DE OBJETOS

IV.1 - Introdução

O Modelo de Representação de Objetos (MRO) representa os objetos de um sistema alvo a ser modelado, os relacionamentos que existem entre os objetos, e identifica os atributos tanto de objetos quanto de relacionamentos, de uma maneira semelhante a do Modelo Entidade-Relacionamento (ME-R). No entanto, uma vez efetuado o modelamento, uma base de dados usando o ME-R armazena os dados em forma de Relações (Tabelas) de Entidades e de Relacionamentos, enquanto isso não ocorre no MRO.

O MRO, devido às características das aplicações a que se destina, tem uma quantidade de Tipos de Objetos, Tipos de Atributos e Tipos de Relacionamentos muito maior, sendo a ocorrência de cada um deles muito menos freqüente do que quando comparado com uma aplicação típica do Modelo Relacional ou do Modelo Entidade-Relacionamento, no sentido em que se fossem usadas tabelas para armazenar cada tipo de objeto ou de relacionamento no MRO, estas seriam muito menores, tendo comparativamente muito menos regis-

tros do que comumente ocorre nos outros modelos.

Além disso, devido à necessidade de se poder modificar o esquema de dados de uma aplicação depois que a Base de Dados correspondente já tenha sido inicializada, a flexibilidade necessária de se poder associar atributos e relacionamentos a cada entidade, e atributos a cada relacionamento, de maneira praticamente individualizada, faz com que não seja adequado o emprego de tabelas, como uma maneira de organizar a estrutura de armazenagem de dados de uma base de dados que usa o MRD.

Como será visto a seguir, é mais conveniente que, para a Estrutura de Dados Independente do Caminho de Acesso do Modelo de Representação de Objetos adote-se um modelo de armazenagem de dígrafos [HAR_69] ao invés da armazenagem direta das tabelas de dados da aplicação, uma vez que a própria estrutura do Esquema de Dados constitui-se em um dígrafo, e pode levar em conta a individualidade da estrutura de cada objeto que ocorre no sistema alvo.

Como consequência da muito mais extensa capacidade de modelagem provida por um dígrafo do que por uma tabela, existe um correspondente aumento das funções e operações que podem ser levadas a efeito sobre dados armazenados em forma de dígrafo do que em forma de tabelas. Dessa forma, as operações que em uma Base de Dados apoiada no Modelo Entidade-Relacionamento poderiam corresponder apenas às oriundas das operações sobre tabelas, ou correspondentes à Álgebra das Relações, no Modelo de Representação de Objetos elas serão estendidas com uma nova gama de funções que tipicamente podem ser encontradas na manipulação de dígrafos.

Neste capítulo serão apresentadas especificações de estruturas de dados que permitem a armazenagem dos dados de uma base que adota o MRD, e um conjunto de operações de manipulação de dados

que permite aproveitar todas as características das quais o MRO é dotado. Será então apresentada uma estrutura lógica da organização de uma Base de Dados que use o MRO, de forma a identificar como as operações de manipulação de dados e a especificação e uso da semântica dos dados podem ser executadas.

IV:2 - A Estrutura de Dados Independente do Caminho de Acesso no Modelo de Representação de Objetos.

Conforme foi visto no capítulo anterior, os dados de uma Base de Dados que utiliza o MRO são armazenados em vários arquivos, sendo que além dos arquivos de objetos e de relacionamentos, existem os vários arquivos de atributos em que cada um armazena os atributos, tanto de objetos quanto de relacionamentos, que sejam de uma mesma característica e que tenham o mesmo formato de dados.

Deve ser notado que, ao contrário dos demais modelos de dados, cada arquivo no MRO não armazena os dados de um mesmo item de especificação do esquema, mas sim cada arquivo armazena dados que são estruturalmente semelhantes, mesmo que correspondam logicamente a itens diferentes, e até de estruturas diferentes. Essa característica costuma ocorrer nas Estruturas de Dados Dependentes do Caminho de Acesso do Modelo de Redes. No Modelo de Representação de Objetos, essa característica sobe um nível de abstração devido ao fato de que cada item de dado tem existência própria e individual, seja esse item um objeto, um relacionamento ou um atributo.

Na Figura 16 está o esquema genérico da Estrutura de Dados Independente do Caminho de Acesso do MRO, representada através de um Diagrama de Entidades e Relacionamentos. Note-se que esse diagrama expressa como os itens de informação de um Esquema de Dados do MRO podem ser relacionados, constituindo-se assim em um Meta-Esquema de Dados para o modelo.

Na implementação de um Sistema de Gerenciamento de Base de Dados, a cada entidade do esquema exposto na Figura 16 estará associado um arquivo. Dessa forma existirão os seguintes arquivos:

- Arquivo dos Identificadores: armazena todos os identificadores de objetos da Base de Dados, sejam eles os identificadores principais ou sinônimos;
- Arquivo de Objetos: armazena as informações correspondentes a cada objeto da base de dados;
- Arquivo de Classes: armazena os conjuntos de objetos constritos pelos objetos que possuem uma classe de construção;
- Arquivo de Relacionamentos: armazena todos os relacionamentos existentes na base de dados;
- Arquivo de Atributos: armazena o conjunto de todos os atributos de cada objeto ou relacionamento da base de dados;
- Arquivos dos Atributos das várias características: existirá um arquivo específico para cada Característica de Atributo (Propriedade, Comentário, Regra, Posição, Tempo, etc.) e para cada Tipo de Dado do Atributo que se faça necessário.

Figura 16 - Meta-Eschema de Dados do MRO

As várias Características de Atributos são tratadas sempre da mesma maneira, permitindo que quaisquer características sejam incorporadas ao modelo. No entanto, uma vez definidas quais as Características de Atributos deverão ser suportadas por uma dada implementação de um Sistema de Gerenciamento de Base de Dados que siga o MRO, apenas atributos com aquelas características serão reconhecidos e tratados pelo SGBD. Na análise de quais Características de Atributos deverão ser suportados pelo SGBD, os conceitos de encapsulamento de informação e monitores de dados, da teoria dos Tipos de Dados Abstratos poderão ser usados.

Note-se que os atributos com característica de Identificadores, sejam eles os identificadores principais ou qualquer tipo de sinônimo, são tratados de forma diferenciada dos demais atributos. Isso deve-se ao fato de que Atributos com característica de Identificadores são necessários em qualquer implementação de SGBD que siga o MRO, e além disso, os demais atributos podem ser associados tanto a objetos quanto a relacionamentos, porém os Identificadores somente identificam objetos.

Na Figura 16 não estão indicados os atributos intrínsecos de cada entidade. Uma particular implementação de SGBD poderá armazenar atributos intrínsecos em cada entidade, desde que esses atributos sejam universais, ou seja, estejam definidos em todas as ocorrências de cada entidade. Além disso, a atribuição de valores aos atributos intrínsecos deverá sempre ser exclusividade do próprio SGBD. Exemplos de atributos intrínsecos são Data de Criação e/ou Atualização de cada ocorrência da entidade, Usuário responsável por aquela ocorrência, parâmetros que permitam esque-

mas de proteção de acesso, recuperação de consistência física da base em caso de queda do sistema, etc.

Cada identificador será de um determinado tipo, assim existe associado a cada identificador um tipo, que indicará se o sinônimo é um identificador principal, ou sinônimo, e nesse caso qual o tipo de sinônimo, tal como por exemplo Abreviatura, Nome Traduzido em alguma outra língua, referência, etc. Cada identificador, dentro de uma mesma classe de objetos identificará apenas um objeto. Porém, cada objeto pode ser identificado por um identificador principal e por qualquer número de sinônimos.

Cada objeto pode restringir objetos de várias Classes de Objetos, no entanto cada classe será constricta por apenas um objeto em toda a Base de Dados. Cada classe de objetos constrictos por um objeto, somente poderá ser acessada se for permitido esse acesso através de uma operação explícita de abertura da classe, a qual é efetuada se uma condição for satisfeita. As classes podem ser abertas para consulta ou atualização, e dessa forma o controle de acesso em um ambiente multiusuário pode ser realizado.

Cada Classe de Objetos será composta de todos os objetos que são constrictos pelo objeto que abre aquela classe para acesso. Note-se que cada objeto sempre é constricto por exatamente um objeto, a menos de quando o objeto faz parte da classe global, quando então não será constricto por nenhum objeto.

A constrictão dos objetos é feita através de seus identificadores. Dessa forma o acesso aos objetos através de uma identificação explícita somente é possível se a classe que o restringe estiver aberta, ou através de uma identificação do objeto junto com a identificação de cada classe que deva ser sucessivamente acessada para se obter o acesso ao objeto. No entanto, constrin-

gindo apenas os identificadores, o acesso a um objeto através de um relacionamento que o identifique ou o envolva sempre será imediato.

Cada relacionamento envolve dois ou mais objetos, sendo que cada objeto pode estar envolvido em qualquer número de relacionamentos. Além disso um relacionamento pode identificar um objeto. Um objeto deve sempre ser identificado de alguma maneira. Dessa forma, um objeto pode ser identificado por um identificador, seja ele um sinônimo ou um nome principal, ou pode ser identificado por um relacionamento. Podem existir objetos que sejam identificados ao mesmo tempo por identificadores e por um relacionamento. No entanto, enquanto um objeto pode ter quantos identificadores forem necessários, somente um relacionamento poderá identificá-lo.

Todo relacionamento tem um tipo associado, e relacionamentos que envolvem mais de dois objetos poderão ser decompostos em vários relacionamentos que envolvem apenas dois objetos. Assim por exemplo, no sistema descrito na Figura 14 poder-se-ia representar o fato de que uma ATIVIDADE usa um RECURSO para produzir uma dada PLACA-DE-CI. Esse relacionamento envolve três objetos, no entanto pode-se considerar que é composto de dois relacionamentos binários, pois independente de ser para produzir uma PLACA-DE-CI, uma ATIVIDADE usa um RECURSO; e além disso, independente de usar um RECURSO, a ATIVIDADE produz uma PLACA-DE-CI.

Note-se no entanto, que a existência de dois relacionamentos binários não traduz a mesma informação que é expressa por um relacionamento ternário, uma vez que uma ATIVIDADE poderá usar um certo RECURSO R1 para produzir determinada PLACA-DE-CI C1, ao passo que usará um outro RECURSO R2 para produzir outra PLACA-DE-

CI C2. Dispondo apenas de relacionamentos binários, não ficaria caracterizado qual RECURSO é necessário para produzir cada PLACA-DE-CI, informação essa contida no relacionamento ternário.

No entanto, não pode ser desprezado o fato que independente da existência de relacionamentos múltiplos, os relacionamentos binários que compõem os relacionamentos múltiplos devem ser considerados, uma vez que efetivamente contêm aquela informação. Por exemplo, mesmo que uma ATIVIDADE use um RECURSO sem que com isso produza uma PLACA-DE-CI, ou seja, apenas um relacionamento binário tem lugar, ainda assim o RECURSO é usado. Se em uma determinada aplicação for necessário verificar que ATIVIDADES usam um determinado RECURSO, tanto os relacionamentos binários quanto os múltiplos contribuirão igualmente para completar essa informação. Dessa forma, é importante ressaltar que a estrutura de dados que armazena os relacionamentos de qualquer multiplicidade devem ser todas idênticas, tanto para facilitar a operação de busca de relacionamentos, quanto para impedir a existência de inconsistência e redundância de informações armazenadas na Base de Dados.

Tanto objetos como relacionamentos podem ter atributos, e dessa forma a estrutura de dados que é usada para armazenar os atributos é a mesma para as duas entidades. As Propriedades (os atributos que não tem nenhuma característica especial) têm armazenado o seu tipo e o valor associado. Em um comentário, o "valor" corresponde a um texto, assim a estrutura de dados usada para armazenar comentários é adaptada para essa característica, por exemplo admitindo que muitas palavras, linhas e páginas possam ser armazenadas, e reconhecendo operações de edição de texto, estruturas e operações essas que apenas dizem respeito a atributos com característica de comentário, e não aos demais

atributos.

Atributos com características de posição devem ser armazenados em estruturas de dados que permitam a especificação de posições (talvez cartesianas ou polares), especificação de ângulos e escala, etc., bem como se reconhecem operações próprias, tais como translação e rotação. Já atributos com características de Regra terão armazenados, além do seu tipo, as condições e ações que constituem a regra, bem como serão reconhecidas as operações próprias de regras.

E assim por diante, cada Característica de Atributos deverá ser identificada e integrada ao Meta-esquema de dados de cada implementação particular, bem como esta deverá reconhecer as operações típicas dos dados com as características representadas.

IV.3 - Estrutura Lógica de um Sistema de Gerenciamento de Base de Dados Apoiado no MRD.

Uma consequência imediata da existência de um Meta-Esquema de Dados, que permite definir o esquema de dados de uma dada aplicação, é que o Sistema de Gerenciamento de Dados sempre permitirá a existência de um Meta-Sistema de Gerenciamento de Base de Dados (Meta-SGBD), o qual por sua vez manterá uma Meta-Base de Dados. Na Figura 17 está um diagrama mostrando a estrutura geral de um Sistema de Gerenciamento de Dados apoiado no MRD.

Um Sistema de Gerenciamento de Dados que abrange um Meta-SGBD será tratado neste trabalho como um Meta-Sistema de Geren-

ciamento de Dados, ou Meta-SGD. Em um sistema desses, existem na realidade uma Meta-Base de Dados e uma Base de Dados. A Meta-Base de Dados armazena as informações léxicas, sintáticas e semânticas que constituem o Esquema de dados das informações manipuladas pela Base de Dados, ou seja, as informações que especificam como os dados se relacionam e qual o seu significado na Base de Dados.

O Meta-Sistema de Gerenciamento de Dados constitui-se na realidade do Gerenciador do Esquema de Dados, armazenado na Meta-Base de Dados. Esse Esquema de Dados incorpora as informações de Sintaxe da Linguagem de Definição de Dados da Base de Dados, e nesse sentido apresenta-se como um Esquema de Dados semelhante aos usados nos demais modelos. No entanto deve ser ressaltado que, sendo definido pelo Meta-Esquema, o esquema de uma aplicação armazena também as informações sobre os próprios dados, e mais que isso, permite que essas informações sejam representadas em uma estrutura sintática idêntica à que é usada para representar os próprios dados. Esse é o motivo que permite que o próprio Meta-Esquema do modelo MRO seja representado usando um Diagrama de Entidades e Relacionamentos, tal como é mostrado na Figura 16.

A Meta-Base de Dados é usada pelo SGBD para manipular os dados da Base de Dados. Além disso, tendo a mesma estrutura de uma Base de Dados, a Meta-Base de Dados pode ser manipulada usando as mesmas ferramentas de Software que permitem a Manipulação de uma Base de Dados. O que pode-se perceber de imediato com esse enfoque, é que uma aplicação pode tanto manipular uma Base de Dados quanto manipular a própria definição dessa mesma Base. O que deve ser verificado é o quanto uma alteração na Meta-Base de Dados afeta a própria Base de Dados. Esse detalhe será analisado a seguir.

Figura 17 - Estrutura Lógica de um sistema usando MRO

Os dados que constituem a informação semântica armazenada na Meta-Base de Dados estão representados na Figura 17, separados em duas partes: Semântica Básica e Semântica Avançada. Isso deve-se ao fato de que, ao contrário da Sintaxe, que pode ser totalmente especificada, a semântica diz respeito ao quanto pode-se representar do significado dos dados, e esse significado não tem um quantificador absoluto que possa caracterizar um total ou uma porcentagem do total. Apenas pode-se representar mais significado, ou menos significado, mas não é possível ponderá-lo.

Além disso, quanto maior a quantidade de informação armaze-

nada e/ou tratada, maior a complexidade do sistema de apoio que manipula essa informação. Os esquemas de dados dos modelos de rede e hierárquico representam praticamente apenas a sintaxe dos dados. Já os modelos Relacional e Entidade-Relacionamento permitem representar uma quantidade semântica bastante maior.

A semântica que pode ser representada nos modelos Relacional e Entidade-Relacionamento é a que corresponde em geral às propriedades matemáticas básicas da álgebra de relações, tais como simetria de relacionamentos, transitividade e as relações derivadas dessas, além de uma cardinalidade estendida, que permite indicar o mínimo e máximo de cada relacionamento, bem como estabelecer esses números a partir da existência ou forma de outros relacionamentos, tais como exclusão mútua ou implicação de outros relacionamentos (conseqüentes) a partir de relacionamentos antecedentes. Esse nível de significado é o que se considera ser a Semântica Básica, e deve ser sempre especificada, em qualquer Esquema de dados de uma aplicação.

A Semântica Avançada representa informações mais refinadas sobre a Base de Dados, e pode-se pensar que constitui um caminho aberto para uma maior densidade semântica no modelo, conforme uma maior capacidade de representação for sendo estruturada. Pode-se considerar informação suprida como semântica avançada o refinamento da semântica básica.

Isso é conseguido admitindo que os tipos de objetos e os tipos de relacionamentos podem ser refinados em subtipos, e estes novamente refinados em novos subtipos, ao mesmo tempo que os subtipos são reagrupados formando outros subtipos, tipos e supertipos, os quais relacionam-se usando novamente as mesmas propriedades usadas para a descrição da semântica básica. Para

efeito da apresentação neste trabalho, será considerado que os tipos sempre constituem a informação semântica básica, enquanto os subtipos e supertipos ocorrem apenas quando refere-se à semântica avançada, sendo que esta não introduz nenhum novo tipo. Note-se que tanto tipos de objetos quanto de relacionamentos possuem subtipos, porém apenas tipos de objetos possuem supertipos.

Por exemplo, na aplicação cujo modelo constitui a Figura 14 poder-se-ia considerar que os objetos do tipo RECURSOS fossem na realidade o agrupamento de objetos de vários outros tipos que seriam também RECURSOS, porém com propriedades adicionais. Por exemplo, poderiam existir recursos-de-imóveis, recursos-de-equipamento-eletrônico, recursos-de-mecânica, que na realidade constituiriam os subtipos IMÓVEL, EQUIPAMENTO-ELETRÔNICO, FERRAMENTARIA, etc.

Cada um desses subtipos constituem-se na realidade de um RECURSO, e dessa forma mantêm os mesmos atributos, relacionamentos, etc. que são aplicados ao tipo RECURSOS, porém cada um contém características próprias, significados que são explicados através de atributos e relacionamentos cuja descrição segue os mesmos padrões assumidos para a semântica básica, porém que a refina, através do refinamento dos tipos de que trata.

O problema associado à manipulação da semântica avançada torna-se assim menos de gerenciamento de estruturas complexas do que do problema de gerenciamento de um volume muito maior de informações, necessitando de um potencial computacional muito mais elevado. Essa é a razão de manter no modelo uma separação entre a semântica que envolve os subtipos, separada daquela que trata apenas dos tipos fundamentais que existem em uma dada

aplicação.

Feita essa separação, se o sistema de gerenciamento de dados reproduzir nos seus procedimentos essa mesma separação, pode-se fazer o tratamento das operações que envolvem os subtipos separadamente daquelas que não os envolvem, e pode ser deixado assim a critério da aplicação, ou do usuário que a estiver usando, a decisão sobre quando efetuar as operações sem usar os significados mais detalhados (e assim obter a operação mais rápida do sistema), e quando efetuar as operações levando em conta um maior "conhecimento" sobre os dados manipulados (e assim empregando necessariamente uma maior quantidade de recursos computacionais).

IV.3.1. - Como é Efetuada a Descrição do Esquema de Dados.

Para que o Meta-Sistema de Gerenciamento de Dados possa ser usado por uma aplicação, o Esquema de Dados deve ser definido, e a Meta-Base de Dados carregada. Essa operação é efetuada fazendo-se a descrição dos elementos léxicos, da sintaxe e da semântica contida no esquema de dados. Com isso, o sistema passa a dispor da Linguagem de Manipulação de Dados, tal como é descrita por esses elementos.

Os elementos léxicos dessa linguagem são obtidos através do levantamento de:

- todos os tipos básicos de objetos;
- todas as classes;
- todos os tipos básicos de relacionamentos;
- todos os tipos de atributos.

A descrição da sintaxe é feita através da especificação de:

- quais tipos de objetos compõem cada classe;
- como cada classe é aberta;
- quais tipos de relacionamentos são permitidos entre cada tipo de objeto;
- quais os tipos de relacionamentos binários que compõem cada tipo de relacionamento múltiplo;
- quais as características de cada tipo de atributo;
- qual a máscara de dados associada a cada tipo de propriedade;
- quais tipos de sinônimos são permitidos a cada tipo de objeto;
- quais tipos de atributos são permitidos a cada tipo de objeto e a cada tipo de relacionamento.

A descrição semântica é efetuada em duas partes separadas. Inicialmente toda a semântica básica é descrita. Essa semântica é constituída da especificação de:

- cardinalidade mínima e máxima de todos os tipos de relacionamentos;
- indicação de se um tipo de atributo pode ter apenas um ou vários valores;
- indicação das relações existentes entre os tipos de relacionamentos (simetria, transitividade);
- indicação de quais tipos de relacionamentos constituem-se em relações de equivalência;
- indicação de exclusão mútua entre tipos de relacionamentos, e entre tipos de atributos;

- indicação de implicação de existência e cardinalidade de relacionamentos dependendo da existência de outros relacionamentos em objetos de cada tipo.

Para efetuar-se a descrição da semântica avançada, é necessário um levantamento dos novos elementos léxicos existentes. Esses elementos são constituídos por todos os subtipos tanto de objetos quanto de relacionamentos, e por todos os supertipos de objetos. A seguir, a sintaxe desses novos elementos léxicos deve ser especificada, e essa operação envolve a descrição de:

- quais os subtipos de objetos compõem cada tipo de objeto e cada subtipo de objeto;
- quais os subtipos e tipos de objetos compõem cada supertipo de objeto;
- quais subtipos de relacionamentos compõem cada tipo de relacionamento e cada subtipo de relacionamento;
- quais subtipos de relacionamentos são permitidos entre cada tipo, subtipo e supertipo de objeto;
- quais tipos de sinônimos são permitidos a cada subtipo e supertipo de objeto;
- quais tipos de atributos são permitidos a cada subtipo e supertipo de objeto e a cada subtipo de relacionamento.

Uma vez efetuada a descrição da sintaxe dos novos elementos léxicos, necessários à semântica avançada, a descrição da semântica avançada propriamente dita pode ser efetuada. O detalhamento das informações que podem ser descritas será mais profundamente analisada na seção IV.5. Sinteticamente, essa descrição é feita

através da especificação das mesmas propriedades descritas para a semântica básica, acrescida de:

- uma regra de derivação, que indica quando cada objeto de um determinado tipo ou subtipo original, é de um particular subtipo que compõe aquele tipo ou subtipo original;
- especificação de conexões lógicas associadas aos subtipos e supertipos de objetos e relacionamentos, tais como intersecção e união.

Uma vez terminada a descrição de um esquema de dados, a base de dados pode ser usada. Para a manipulação da base, o SGBD é acionado, e as operações típicas de inserção, atualização e consulta de dados poderão ser efetuadas.

IV.4. - Operações Básicas de Manipulação de Dados da Base de Dados.

O Sistema de Gerenciamento da Base de Dados, que opera segundo o Modelo de Representação de Objetos, será analisado através da decomposição de seus módulos em quatro níveis lógicos:

- acesso aos dados da Meta-Base de dados;
- operações básicas de manipulação dos dados da Base de Dados;
- operações de manipulação dos dados da base de dados levando em conta a semântica básica do esquema;
- operações de manipulação dos dados da Base de Dados

levando em conta toda a semântica do esquema.

O SGBD acessa a Meta-Base de dados apenas para consulta, assim os módulos do primeiro nível constituem-se de módulos de obtenção de dados da Meta-base de dados, os quais permitem aos módulos dos outros níveis o acesso a essas informações. Esses módulos constituem-se basicamente em rotinas de acesso à informação armazenada em forma de digrafo, que é como a Meta-Base de Dados armazena as suas informações.

As rotinas desse nível consultam a Meta-Base de Dados para obter as informações léxicas e sintáticas da linguagem lá definida. As rotinas executam as seguintes tarefas:

- Verificar se uma dada palavra é um símbolo da linguagem;
- Obter o símbolo da linguagem que corresponde a determinado tipo de um elemento da base de dados;
- Obter os tipos de objetos que compõem cada classe;
- Obter a especificação de que condições devem ser satisfeitas para abrir uma classe;
- Verificar se determinado tipo de relacionamento é válido entre determinados tipos de objetos;
- Verificar quais os tipos de relacionamentos binários compõem determinado tipo de relacionamento múltiplo;
- Obter as características de determinado tipo de atributo;
- Obter a máscara de dados associada a determinado tipo de propriedade, bem como as informações essenciais necessárias para a manipulação de atributos de cada característica, tais como por exemplo, indicação do conjunto de tipos de traços e tipos de preenchimento

- de áreas para atributos com característica de gráficos;
- Verificar se é permitida a existência de um determinado tipo de sinônimo para um determinado tipo de objeto;
 - Verificar se determinado tipo de atributo (que não tenha característica de sinônimo) é permitido a determinado tipo de objeto ou tipo de relacionamento.

Os módulos dos dois últimos níveis serão tratados na seção IV.5, e usam as operações que são definidas nos dois módulos anteriores.

Os módulos do segundo nível, os que realizam as operações básicas de manipulação de dados da Base de Dados são os que efetivamente realizam as operações de colocação de dados, alterações e eliminação de dados e consulta à base de dados. Sendo operações básicas, as rotinas que realizam essas funções não levam em conta nem a sintaxe e nem a semântica particular de nenhum esquema de dados, sendo portanto subordinadas aos módulos dos dois níveis superiores, que efetivamente realizam a verificação de validade de cada particular operação solicitada.

A descrição de um conjunto completo de rotinas, que efetuem todas as operações necessárias em um determinado SGBD, fica sujeita ao conjunto de Característica de Atributos escolhido para aquele SGBD. Nas discussões seguintes supõe-se que além de propriedades e sinônimos, apenas atributos com característica de comentário são tratados pelo SGBD. As rotinas necessárias ao suporte de outras Características de Atributos podem ser inferidas diretamente das operações básicas necessárias à determinada característica e seguir o mesmo padrão mostrado para a característica Comentário.

Em um SGBD que trate de atributos com as características indicadas, os vários elementos que compõem uma Base de Dados podem ser vistos na Tabela 1 (como mostra o diagrama da Figura 16).

- 1) Classes de Objetos
- 2) Objetos
- 3) Relacionamentos Binários
- 4) Relacionamentos Múltiplos
- 5) Sinônimos
- 6) Comentários de Objeto
- 7) Comentários de Relacionamento
- 8) Propriedades de Objeto
- 9) Propriedades de Relacionamento

Tabela 1 - Elementos de uma Meta-Base de Dados

Para efeito da escolha de operações permitidas com um determinado atributo, a sua característica deve ser determinada. Assim, para que se efetue uma operação típica de comentário, o atributo envolvido deve ser da característica Comentário. No entanto, para se fazer referência ao atributo, apenas a referência ao seu tipo identifica a sua característica. Por essa razão, nas rotinas apresentadas a seguir, comentários, sinônimos etc. são referenciados apenas pelo seu código de atributo.

Os vários elementos de uma base de dados são acessados através da indicação de um identificador de um objeto, e a partir

dai, os demais elementos são acessados de acordo com a forma como se associam ao objeto identificado. O acesso às informações de uma base de dados apoiada no MRD é dessa forma essencialmente através de navegação. Isso significa que a base de dados é acessada deslocando-se um Ponteiro virtual sobre a base. Sempre existirá um ponteiro indicando qual é o objeto corrente, qual é o relacionamento corrente, qual o comentário de objeto corrente, etc. Dessa forma, para qualquer operação a ser realizada na Base de Dados, deve-se primeiro posicionar os ponteiros adequadamente e a seguir efetuar-se a operação desejada. Por outro lado, é necessário que se tome conhecimento das posições apontadas antes de se realizar qualquer operação sobre a base de dados.

As rotinas que operam sobre cada elemento da base de dados recebem e fornecem parâmetros que especificam cada execução. Para a apresentação das rotinas, serão usadas as abreviaturas indicadas na tabela 2.

CO	- Código de Objeto = referência direta à um objeto;
CTO	- Código de Tipo de Objeto;
COC	- Código do Objeto que constringe uma Classe;
CTR	- Código de Tipo de Relacionamento;
CTP	- Código de Tipo de Propriedade;
OC	- Objeto Corrente;

Tabela 2 - Abreviaturas usadas nos parâmetros de acesso à Base de Dados.

A menos das classes existentes em uma base de dados, os demais elementos, em geral, podem ter definidas as seguintes

operações:

- a) Colocar novos dados;
- b) Modificar dados já existentes;
- c) Posicionar para as operações subsequentes;
- d) Consultar os dados existentes;
- e) Retirar dados já existentes;
- f) Outras operações variadas.

Nas seções seguintes, é descrito um conjunto mínimo de operações básicas que o SGBD deve suprir em termos de manipulação dos dados em uma base de dados, e que constitui-se então no conjunto mínimo de operações de manipulação de dados de uma Base de Dados.

IV.4.1 - Operações de Apoio às Funções de Manipulação de dados.

Além das funções básicas de manipulação de dados da base, que envolvem os elementos que constituem a própria base de dados, outras funções são necessárias para facilitar, ou mesmo permitir que toda a força do modelo possa ser atingida. Cada conjunto de funções expostas a seguir se apoiam em um conceito que é explicado em cada uma das subseções seguintes.

IV.4.1.1. - Sobre o posicionamento de apontadores: A Marcação de Estado.

Conforme foi explicado anteriormente, cada elemento que compõe uma base de dados tem um apontador, que indica qual é a instância daquele elemento considerado corrente num dado instante. Segundo esse conceito, é possível ter ao mesmo tempo um objeto corrente, ter um de seus relacionamentos como relacionamento corrente, e ter também um atributo de cada característica associado ao objeto corrente como sinônimo, comentário e propriedade corrente, bem como ter um atributo de cada característica associado ao relacionamento corrente, tal como comentário de relacionamento corrente e propriedade de relacionamento corrente.

Note-se no entanto que nem sempre todos esses apontadores estarão definidos. São Reconhecidos então três níveis de apontadores:

- Apontador para o objeto corrente;
- Apontador para o relacionamento corrente e apontadores para os atributos de objeto correntes;
- Apontadores para os atributos de relacionamento correntes.

Note-se que o apontador de relacionamento corrente e os apontadores de atributos de objeto correntes apontam sempre para um relacionamento e para atributos associados ao objeto corrente. Dessa forma somente estarão definidos se forem estabelecidos após a última mudança do objeto corrente, pois sempre que este muda, os apontadores que dele dependem tornam-se indefinidos, e somente voltam a ser definidos após uma operação que os defina explicitamente.

O mesmo ocorre com os apontadores para os atributos de relacionamento, que tornam-se indefinidos quando o apontador de relacionamento corrente muda ou se torna indefinido.

As operações que definem e movimentam os apontadores são as operações básicas sobre as entidades da base de dados, notadamente as operações de posicionamento. No entanto, dessa forma apenas um ponteiro existe para cada entidade da base. Para permitir uma multiplicidade de ponteiros para cada entidade o conceito de "Marcação de Estado" deve ser introduzido.

Esse conceito parte do pressuposto que existe um conjunto de elementos de informação que constitui uma base de dados. No caso do SGBD que está sendo usado para esta apresentação, esse conjunto é composto pelos 9 elementos que aparecem na Tabela 1, com exceção de Classe. O "Estado Atual" de uma base de dados, com relação ao acesso que esteja sendo feito por uma determinada aplicação, consiste no conjunto de todos os apontadores, esteja cada um deles definido ou não, e no conjunto de todas as classes abertas.

Se num dado instante o conjunto de todos os apontadores for "fotografado" e armazenado, e depois os apontadores forem alterados, é possível retornar ao estado "fotografado" simplesmente reposicionando os apontadores para o estado armazenado. Esse constitui o conceito de "Marcação de Estado".

Mais de um estado pode ser marcado, permitindo uma multiplicidade de estados. Os vários estados marcados podem ser estruturados da maneira que for julgado conveniente quando o SGBD for planejado. No entanto, basicamente dois tipos de estrutura devem ser suportados: Indicação Individual e Pilha.

A Indicação Individual de Estado permite que se atribuam

índices a cada estado marcado ou reposicionado, permitindo que se disponha de um número qualquer de ponteiros, que podem ser individualmente acessados e alterados.

A manutenção de uma Pilha de Estados permite que se efetuem sobre a base de dados operações recursivas, operações essas bastante comuns em operações de navegação complexas na Base de Dados.

Para aumentar a flexibilidade das operações associadas à Marcação de Estado, pode-se adicionar ao conjunto de apontadores mais alguns campos de dados de usuário, que são deixados para uso do próprio aplicativo. Assim, além de marcar um estado e para ele poder retornar, o aplicativo pode ter também um conjunto de parâmetros que a seu critério podem estar associados a cada estado marcado. O número de campos de dados de usuário existentes deve ser definido na particular implementação de cada SGBD.

As rotinas básicas que implementam o conceito de "Marcação de Estado" em um Base de Dados apoiada no MRO são as seguintes:

- Inicializar a estrutura de marcação de estado. Deve ser indicado o número máximo de estados que poderão ser armazenados, e a forma como a estrutura será usada, se como pilha ou através de Indicação Individual. A inicialização deve ser feita antes de se executar as outras rotinas que implementam este conceito, e podem ser feitas reinicializações sempre que os estados já armazenados não sejam mais necessários.
- Armazenar um valor num campo de dado de usuário.
- Obter o valor armazenado num campo de dado de usuário.
- Empilhar o estado atual. O estado atual é colocado na pilha, mas não é alterado. Somente pode ser executada se a estrutura de marcação estiver sendo usada como pilha.

- Desempilhar. O estado atual é substituído pelo estado armazenado no topo da pilha, e este é retirado da pilha.
- Armazenar o estado atual com um índice indicado.
- Reposicionar no estado indicado por um índice.

IV.4.1.2. - Seleção de Objetos

No Modelo de Representação de Objetos, as operações básicas acessam cada objeto, cada relacionamento, ou cada atributo individualmente, caracterizando a unicidade com que podem ser acessados cada um dos objetos que modelam o sistema alvo de uma aplicação. No entanto, conjuntos de objetos podem ser obtidos através de seleção de objetos. Sobre um ou mais conjuntos de objetos selecionados, as operações usualmente empregadas nos modelos Relacional e Entidade-Relacionamento podem ser executadas.

As rotinas básicas que implementam as operações de seleção de objetos são as seguintes:

- Abrir um conjunto de seleção de objetos, para operações posteriores de inclusão ou eliminação de objetos.
- Criar um conjunto de seleção de objetos contendo os objetos da base de dados que satisfaçam um critério de seleção especificado. O critério de seleção envolve a existência de determinado relacionamento ou atributo, ou determinado tipo de objeto, de toda a base ou de uma classe cujo objeto que a restringe é especificado ou é o Objeto Corrente (OC), ou da classe desse tipo aberta.
- Indicar que devem ser armazenados em determinado conjunto de

seleção já aberto os objetos que em operações subsequentes forem criados e/ou modificados em uma determinada classe constricta por um objeto especificado ou pelo OC, ou em qualquer classe. Se o conjunto de seleção indicado já tivesse indicação semelhante, a nova especificação de quais objetos selecionar seria incluída com as já existentes.

- Retirar de um conjunto de seleção uma indicação de inclusão de objetos especificada pela rotina anterior.
- Fechar um conjunto de seleção de objetos especificado.
- Eliminar um conjunto de seleção de objetos especificado.
- Esvaziar um conjunto de seleção de objetos especificado.
- Verificar se um determinado objeto está no conjunto de seleção de objetos especificado, o qual deve estar aberto.
- Colocar um determinado objeto no conjunto de seleção de objetos especificado, o qual deve estar aberto.
- Retirar um determinado objeto de um conjunto de seleção de objetos especificado, o qual deve estar aberto.

Uma base de dados pode conter uma quantidade muito grande de objetos, mesmo de objetos que possam ser selecionados para uso em uma dada aplicação. Pode ser necessário que a aplicação necessite atribuir valores temporários a cada objeto de uma seleção, para efeito dos procedimentos que estiver executando. Dessa forma, seria conveniente que o SGBD permitisse a atribuição de valores individualmente a cada objeto de uma seleção.

Se os valores que se deseja especificar corresponderem a um tipo de propriedade prevista no esquema da base de dados, essa atribuição pode ser feita usando-se as rotinas usuais de atribuição de valores a propriedades. No entanto, existem situações em

que essa técnica não é aconselhável, principalmente devido ao fato de que esses valores devem ter validade somente durante o período de execução do procedimento, não tendo nenhum significado fora dele.

Para suprir essa necessidade, podem ser desenvolvidas rotinas que atribuem valores aos objetos de uma seleção, mantendo esses valores associados à seleção, e não à base. Os valores teriam então a duração da existência da seleção onde estiverem associados. Note-se que o objetivo desse recurso é atribuir valores numéricos, que servirão para a execução de procedimentos dos programas de aplicação, e não para atribuir valores a atributos de qualquer das características especiais que o SGBD suporte. Essa última operação deverá ser efetuada apenas usando as rotinas usuais de manipulação de atributos.

As rotinas que desempenham a função de associar valores aos objetos de um conjunto de seleção de objetos são as seguintes:

- Atribuir um valor inicial especificado a todos os objetos de uma seleção. Objetos posteriormente incluídos nessa seleção também terão associado esse mesmo valor.
- Atribuir um valor a um objeto especificado de um conjunto de seleção especificado.
- Consultar o valor especificado para um determinado objeto de um conjunto de seleção especificado.

IV.4.1.3. - Relacionamentos e Atributos Automáticos

Cada particular implementação de um Sistema de Gerenciamento de Bases de Dados determinará quais são os atributos intrínsecos que integrarão os registros físicos que irão constituir a base de dados, e esses registros são de inteira responsabilidade do próprio SGBD. No entanto, uma aplicação poderá necessitar que o SGBD assuma também o controle de outros atributos ou relacionamentos que não são intrínsecos, mas são especificados pelo aplicativo para serem gerenciados pelo SGBD, tal como incluir determinado atributo ou relacionamento a cada novo objeto que for criado ou acessado. Essas operações que são feitas pelo SGBD sob solicitação do aplicativo são denominadas de "Relacionamentos e Atributos Automáticos".

A indicação de um relacionamento automático é feita pelo tipo do relacionamento e do objeto destino. A indicação de um atributo, podendo ser ele de qualquer característica, é feita estabelecendo-se um objeto fictício para o qual o atributo é definido, através de seu tipo e valor.

A colocação dos relacionamentos e atributos é efetuada pelas rotinas básicas de manipulação de dados da base de dados, e somente será efetivada se o atributo ou relacionamento for sintaticamente válido para o tipo de objeto ou de relacionamento que deverá receber a colocação automática. A indicação de quais atributos e relacionamentos devem ser colocados automaticamente é efetuada pelas seguintes rotinas:

- Estabelecer um objeto fictício para a determinação de um atributo. O objeto fictício torna-se o Objeto Corrente, e sobre ele é definido o atributo, de qualquer característica,

usando as operações usuais de colocar atributos. Quando um outro objeto for marcado como corrente, o objeto fictício estabelecido deixa de existir.

- Indicar um relacionamento binário a ser colocado em todo objeto criado ou modificado. A abrangência da indicação é especificada como os objetos constrictos por um objeto especificado, pelo objeto corrente ou como todos os objetos da base de dados.
- Indicar um atributo a ser colocado em todo objeto criado ou modificado. A abrangência da indicação é feita da mesma maneira que para a rotina anterior.
- Indicar um atributo a ser colocado em todo relacionamento criado ou modificado.
- Cancelar qualquer uma das indicações anteriores.

Quanto ao conjunto completo das possíveis indicações de relacionamentos automáticos, pode ser mencionado o caso da indicação de relacionamentos múltiplos, e o caso da indicação de se estender automaticamente um relacionamento explícito para envolver mais objetos. Esses dois casos não são tratados aqui, e nem as rotinas descritas acima são suficientes para cobri-los, pois a quantidade de alternativas possíveis é muito grande, e a escolha de uma delas depende de situações específicas. Uma explicitação de rotinas que atendessem a todas as possibilidades dependem de parâmetros que estarão envolvidos em qualquer aplicação, o qual é bastante elevado, bem como o são as consequências que devem ser previstas para seu tratamento. Devido a essa complexidade, e ao fato de se tratarem de situações relativamente pouco frequentes,

não foi considerado conveniente a inclusão desses recursos no conjunto de rotinas descritas.

IV.4.2 - Operações básicas sobre Classes.

Cada classe de identificadores de objetos é sempre constrita por exatamente um objeto, e não tem uma identificação própria, a qual é sempre feita através do objeto que a constringe. Dessa forma a referência a uma determinada classe sempre será feita através da referência a um objeto.

Deve ser lembrado aqui o fato de que podem ser abertas várias classes simultaneamente, porém somente uma classe de cada tipo de cada vez. Além disso como os tipos de classes estabelecem uma partição sobre o Conjunto de Tipos de Objetos, sempre que for indicado um tipo de objeto, mesmo que exista mais de uma classe aberta, a classe ao qual o tipo de objeto pertence estará estabelecida. Esse fato vale tanto para as operações básicas sobre classes quanto para as operações que envolvam objetos.

Uma consequência desse fato é que ao abrir-se o conjunto de classes constritas por um objeto, se alguma das classes abertas já tiver outra classe do mesmo tipo aberta, a classe anteriormente aberta deve ser fechada antes de nova abertura da classe daquele tipo.

As operações efetuadas sobre uma classe são:

- Abrir uma classe, constrita por um objeto CO especificado ou pelo objeto corrente (OC). Para a abertura, uma condição de validade deve ser especificada.
- Fechar a classe constrita por um objeto CO especificado ou pelo

Objeto Corrente.

- Obter o estado e o código de proteção atual da classe constrita pelo objeto CO especificado, ou pelo objeto OC.

IV.4.3. - Operações Básicas sobre Objetos.

Os objetos constituem-se no núcleo lógico de todas as operações que devem ser realizadas em uma Base de Dados. Enquanto os demais elementos de uma base de dados somente são acessados através dos ponteiros que indicam qual é o particular elemento corrente, os objetos podem ser acessados diretamente através de um identificador. Para efeito da apresentação que segue, a indicação do identificador principal de um objeto é denominada de "Nome" do objeto.

Um conjunto mínimo de operações que permitem a colocação de objetos na base de dados, aproveitando toda a flexibilidade do Modelo de Representação de Objetos é constituído das operações seguintes. Sempre considera-se que os objetos colocados tornam-se o Objeto Corrente (OC):

- Colocar um objeto dado seu nome e CTO, na classe corrente correspondente.
- Colocar um objeto dado o seu nome e CTO, na classe constrita por uma entidade cujo COC é especificado.
- Colocar um objeto dado o seu nome e CTO, na classe corrente correspondente, com os respectivos relacionamentos automáticos presentemente ativados.

As operações de modificação de objetos também fazem com que objeto modificado se torne o objeto corrente, e são as seguintes:

- Trocar o nome do objeto especificado, ou do objeto corrente, pelo novo nome especificado.
- Trocar o tipo do objeto especificado, ou do objeto corrente, pelo CTO especificado. Essa função somente poderá ser efetuada se os tipos de atributos e os tipos de relacionamentos já definidos para o objeto indicado forem compatíveis com o novo tipo de objeto especificado.
- Aglutinar dois objetos em apenas um, dados os códigos dos dois objetos, sendo que um deles pode ser o OC. Um dos objetos tem seu nome eliminado da base de dados, enquanto o outro passa a incorporar todos os relacionamentos e atributos originalmente associados ao objeto que é eliminado. Essa função somente poderá ser efetuada se os dois objetos forem do mesmo tipo.

As operações de Posicionamento de Objetos apenas mudam a indicação de qual é o Objeto Corrente atual, e são as seguintes:

- Posicionar em um objeto, dado o tipo do objeto e o valor de um identificador. A procura é efetuada na classe aberta que contém os objetos do tipo indicado. O tipo do identificador não precisa ser especificado, uma vez que para os objetos de uma classe os identificadores indicam univocamente um objeto. Se apenas a classe global estiver aberta, o tipo do objeto pode não ser especificado;
- Posicionar em um objeto, dado o valor de um identificador e o código do objeto que restringe a classe onde o objeto deve ser procurado;

- Posicionar em um objeto, dado o seu código;
- Inicializar uma seqüência ordenada de busca de objetos. A busca será feita entre os objetos constringidos por um objeto indicado. Se esse objeto não for indicado, considera-se a classe global. Pode ser útil indicar a busca apenas dos objetos de um determinado tipo, cuja omissão indica a busca de todos os objetos da classe indicada;

As operações de Obter Dados de Objetos fornecem para uma aplicação os meios de consultar a Base de Dados para obter informações sobre o Objeto Corrente, bem como, em alguns casos, mudar o Objeto Corrente de uma maneira pré-determinada. As rotinas que desempenham essas funções são as seguintes:

- Obter o código do objeto corrente;
- Obter o tipo de um objeto, dado o seu código. Se o Código for omitido, considera o Objeto Corrente;
- Obter o código do objeto corrente e avançar. Essa rotina somente pode ser usada depois que tenha sido feita uma inicialização de seqüência ordenada de busca de objetos. O objeto corrente passa a ser o novo objeto posicionado;
- Obter dados de um objeto indicado. Esta rotina retorna para o programa de aplicação informações sobre os atributos intrínsecos do objeto indicado. Quais são essas informações, depende da particular implementação do SGBD. Podem ser por exemplo data de criação e/ou atualização do objeto, código do usuário ou aplicativo responsável pelo objeto, etc;
- Obter o nome (principal) de um objeto, dado o seu código. Se o código for omitido, considera o OC;
- Obter o código do objeto que constringe o objeto indicado. Se o

objeto não for indicado, considera o DC;

Apenas uma rotina é necessária para retirar um objeto da Base de Dados, uma vez que as demais informações associadas a um objeto são retiradas por rotinas de manipulação dessas informações. A rotina realiza a seguinte operação:

- Retira um objeto da base de dados, dado o seu código. Se o objeto não for indicado, retira o DC. Se houver uma sequência de busca inicializada na classe desse objeto, ela é encerrada. Após a retirada de um objeto, deixa de haver um objeto corrente.

Operações variadas envolvendo primordialmente objetos são realizadas por rotinas que manipulam os objetos de uma forma não coberta pelas outras operações. Podem ser previstas rotinas para desempenhar as seguintes tarefas:

- Atribuir a um objeto ainda sem nome (principal) um nome agora especificado. É indicado também o código do objeto, que se omitido passa a ser considerado o DC. A classe que irá restringir esse nome é a classe atualmente aberta que contém objetos do tipo do objeto indicado. Essa rotina é útil para designar um nome a um objeto que foi criado inicialmente para ser identificado por um relacionamento;
- Idem à anterior, porém indicando o código do objeto que deve restringir o nome indicado;
- Colocar no objeto indicado os relacionamentos e atributos automáticos atualmente indicados, sintaticamente válidos para o objeto indicado. Se o objeto não for indicado, considera o DC;

IV.5. - Operações de Manipulação de Dados em Nivel Semântico.

As operações básicas de manipulação de dados da base de dados descritas na seção anterior, realizam apenas as tarefas de colocação, modificação e consulta da base de dados, sem levar em conta a semântica ou a sintaxe associada a um particular esquema de dados. Constituem-se assim nas operações elementares de acesso à base de dados, assim como as operações de consulta ao esquema se constituem nas operações elementares de acesso à Meta-Base de dados.

Usando esses dois conjuntos de operações, um terceiro pode ser construído que se constituirá de operações de colocação e modificação de dados na Base de Dados, que leva em consideração a sintaxe do modelo sendo manipulado. Esse terceiro conjunto será constituído de operações que desempenham as mesmas funções que as de colocação, modificação e retirada de dados do conjunto de operações básicas de acesso à Base de Dados. No entanto, antes de efetuar a operação solicitada, cada rotina desse novo grupo irá verificar a validade da operação segundo as regras de sintaxe que podem ser obtidas através das rotinas de acesso à Meta-Base de Dados.

Aplicações que usem exclusivamente as rotinas de colocação e modificação desse terceiro conjunto, terão a consistência sintática da Base de Dados garantida.

No entanto, além de seguir as regras de validade sintáticas, uma aplicação deve poder se beneficiar do fato que o SGBD pode reconhecer o significado dos dados manipulados, e realizar automaticamente operações que reconhece serem necessárias. No

entanto, essas operações realizadas automaticamente consomem sistematicamente mais tempo de execução, quanto maior a quantidade de conhecimento semântico envolvido. Além disso, é freqüente ocorrer casos em que o conteúdo da base de dados sendo alterada, passe por fases em que a semântica indica estados inconsistentes, que serão tornados consistentes nas operações subseqüentes. Devido a isso, é conveniente que as operações de manipulação de dados da Base de Dados que envolvam conteúdo semântico sejam divididas em três conjuntos, cujas operações elementares podem ser acessadas diretamente pelos programas de aplicações. Os três conjuntos são os seguintes:

- Operações semânticas fundamentais, que devem ser sempre respeitadas. Como exemplo, pode-se citar o fato que todos os relacionamentos sempre tem um relacionamento oposto correspondente, e assim o oposto de um relacionamento cuja colocação é solicitada deve ser sempre automaticamente colocado também;
- Operações semânticas básicas, que constituem-se em operações que completam uma base de dados com informações que se podem deduzir trivialmente das operações já existentes, mas cuja colocação nem sempre é desejável. Como exemplo pode-se citar o fato de se levar em conta que determinados relacionamentos são transitivos;
- Operações semânticas avançadas, que constituem-se em informações complexas, que podem ou não ser levadas em consideração por uma aplicação. Trata das informações semânticas avançadas definidas no esquema de dados. Como exemplo, pode-se citar a colocação de um objeto indicando-se seu subtipo ao invés de se indicar o seu

tipo.

Na Figura 18 está um esquema mostrando como esses vários conjuntos de operações interagem mutuamente, e como se inter-relacionam com um programa de aplicação.

Figura 18 - Interação entre as operações em um sistema

IV.5.1. - Especificação Semântica Fundamental

As operações de acesso à Base de dados solicitadas pelo programa de aplicação indicam três níveis de conteúdo semântico que devem ser levados em conta quando a operação é executada. Cada informação semântica que é usada reflete-se na operação do Sistema de Gerenciamento de Bases de Dados através da execução de outras operações a mais na Base de Dados, além da explicitamente solicitada pelo aplicativo.

O nível que utiliza a menor quantidade possível de informação semântica deve levar em conta toda a sintaxe do modelo, e as operações que em qualquer modelo mantêm-se constantes. Essas operações são as seguintes:

- Tratamento idêntico para o relacionamento oposto para cada operação de Inclusão, Modificação ou Retirada de relacionamento;
- Identificação dos relacionamentos simples que compõem um relacionamento múltiplo, sempre que uma operação que envolve um relacionamento múltiplo é solicitada;
- Verificar as Anotações para acesso que possam envolver o objeto acessado, sempre que qualquer objeto seja acessado, através de qualquer operação que altere a indicação de qual é o atual objeto corrente;
- Verificar as regras que possam ter sua condição satisfeita sempre que um atributo com característica de regra for acessado, através de qualquer operação de posicionamento e obtenção de dados, ou alteração de objetos.

O tratamento do relacionamento oposto deve ser feito porque

no MRO qualquer relacionamento tem obrigatoriamente um relacionamento oposto. Dessa forma, se um relacionamento é colocado, o seu oposto já estará plenamente definido e sempre deverá existir na base de dados. Sendo colocado automaticamente, a completeza da descrição sob esse aspecto pode ser sempre garantida. Da mesma forma, as operações de modificação e retirada de relacionamentos devem atuar no relacionamento oposto ao indicado da mesma maneira que no relacionamento indicado.

A identificação dos relacionamentos simples que compõem um relacionamento múltiplo deve ser feita para que todas as componentes relevantes de um relacionamento sejam também localizadas e levadas em conta quando um acesso envolvendo apenas parte do relacionamento múltiplo, ou um relacionamento binário é feito. Por exemplo, o fato de um RECURSO ser usado por um PROJETO para desenvolver uma PLACA-DE-CI constitui um relacionamento múltiplo, porém o relacionamento binário componente que indica que um RECURSO é usado por um PROJETO deve ser reconhecido quando os PROJETOS que usam aquele RECURSO forem procurados. Isso é conseguido fazendo-se com que os relacionamentos binários que compõem cada relacionamento múltiplo sejam reconhecidos e indicados na base de dados sempre que uma operação de colocação, modificação ou retirada de relacionamento múltiplo for solicitada.

As operações de verificar as Anotações de Acesso e Regras acessadas são feitas no sentido de que, se uma condição verificada for satisfeita, então a ação correspondente àquela anotação ou regra deve ser executada. A execução de uma ação é constituída pela execução de uma ou mais operações que serão indicadas para operação posterior, tal como foi apresentado no Capítulo III. Os conjuntos de operações postergadas são agrupadas

em transações, cada uma contendo operações provenientes de uma mesma regra ou anotação de acesso.

IV.5.2. - Especificação Semântica Básica.

A descrição semântica é efetuada em duas fases distintas. A primeira fase tem por objetivo a descrição da semântica associada aos elementos léxicos e sintáticos referentes aos tipos de objetos, de relacionamentos e de atributos, sendo que as informações que podem referir-se a generalizações ou especializações de tipos, através de subtipos e supertipos são deixados para a segunda fase, que ocupa-se dessas informações.

As informações semânticas básicas são então os dados armazenados na Meta-Base de Dados que fornecem informações sobre os tipos de objetos, atributos e relacionamentos, e o significado de cada associação que existe entre eles. Para isso, são parametrizadas cada uma das formas com que as ocorrências de qualquer uma dessas entidades se associam. As associações semânticas básicas entre entidades desses vários tipos podem ser divididos em cinco grupos, como mostra a Figura 19.

Nessa forma de associação, sempre que um atributo refere-se a um objeto, consideração idêntica deve ser entendida entre atributo de relacionamento e relacionamento. No entanto, para não tornar a discussão a seguir repetitiva, não será mencionada a segunda, porém deve-se entender que tudo o que for válido para associações entre objeto e atributo, vale igualmente para associações entre relacionamentos e atributos de relacionamentos.

1-informações sobre relacionamentos

2-informações sobre atributos

3-informações sobre associações entre objetos e relacionamentos

4-informações sobre associações entre atributos e relacionamentos

5-informações sobre associações entre atributos e objetos.

Figura 19 - Grupos de informações semânticas.

IV.5.2.1. - Informações sobre Relacionamentos.

As informações semânticas que ocorrem apenas entre relacionamentos, e que podem ser especificadas em um esquema de dados MRD são as seguintes:

-- Indicação de qual é o tipo de relacionamento binário oposto a cada tipo de relacionamento binário -- No MRD, todo o relacionamento que for indicado entre dois determinados objetos O1 e O2 sempre indicará que existe um relacionamento R1 que permite obter O2 a partir de O1, e um relacionamento R2, chamado relacionamento

oposto a R1, que permite obter O1 a partir de O2. Por exemplo, um PROJETO E' COMPOSTO POR uma ATIVIDADE, e uma ATIVIDADE COMPOE um PROJETO.

Todos os relacionamentos binários no MRO envolvem um objeto origem e um objeto destino. Para o relacionamento oposto a cada relacionamento, os objetos origem e destino trocam de papel. Deve ser ressaltado que um relacionamento e o seu oposto são sempre os mesmos, não importando os tipos de objetos origem ou destino a que se associem.

-- Indicação de quais são os tipos de relacionamentos binários simétricos -- Pode ocorrer que o tipo do relacionamento oposto a um dado tipo de relacionamento seja do mesmo tipo. Nesse caso teremos a ocorrência de um relacionamento binário simétrico. Exemplo disso ocorre com a indicação de quais PINOs de cada CIRCUITO-INTEGRADO USADO POR uma PLACA-DE-CI SAO CONECTADOS com quais outros PINOs.

-- Indicação de transitividade para cada tipo de relacionamento binário -- Podem existir na Meta-Base de Dados tipos de relacionamentos que tenham a característica de ser transitivos, ou seja, se um objeto O1 é relacionado através de relacionamento de tipo R1 com outro objeto O2, e o objeto O2 é relacionado com outro objeto O3 através de relacionamento do mesmo tipo R1, então, se o relacionamento do tipo R1 for transitivo, pode-se concluir que ocorre um relacionamento do tipo R1 também entre o objeto O1 e o objeto O3. Um exemplo de tipos de relacionamentos transitivos pode ser o relacionamento do tipo DESCENDENTE DE entre PESSOAS.

-- Tipos de relacionamentos binários que expressam relações de equivalência -- Tendo sido indicado que um determinado tipo de relacionamento é simétrico e transitivo, o MRD assume que aquele tipo de relacionamento constitui uma relação de equivalência. Matematicamente, seria necessário que o relacionamento fosse também reflexivo. No entanto o conceito de relacionamento no MRD torna inexpressivo qualquer relacionamento entre um objeto e ele mesmo, tornando sem sentido um possível conceito de relacionamento reflexivo. Por isso, o termo "Relação de Equivalência" é empregado no MRD para designar agrupamentos de objetos que mantêm entre qualquer par de objetos pertencentes a esse agrupamento o relacionamento assumido como capaz de expressar a relação de equivalência.

Note-se que os objetos envolvidos em uma relação de equivalência não precisam ser todos do mesmo tipo, uma vez que tanto a transitividade quanto a simetria do tipo de relacionamento que constitui a relação de equivalência não pressupõem que o tipo de objeto origem de cada relacionamento seja do mesmo tipo do objeto destino. Assim, conceitos de simetria, transitividade e relações de equivalência no MRD são independentes dos tipos de objeto a que possa estar associado cada relacionamento.

-- Indicação de dependência existencial de objetos a partir de tipos de relacionamentos -- Determinados objetos podem necessitar de outros para que possam ser significativos em uma Base de Dados. Sempre que um objeto precisar de outro, esses dois objetos estarão ligados através de um relacionamento de dependência existencial, que estabelece qual é o objeto responsável pelo objeto

dependente. Todo objeto que for criado na base de dados de um tipo que admita relacionamentos que indica dependência existencial será aceito apenas se o relacionamento for indicado com o respectivo objeto responsável. Todo objeto que for retirado da base de dados que esteja ligado a outros através de relacionamentos opostos a relações de dependência existencial obrigará a que os objetos dependentes sejam também retirados da Base de Dados.

-- Indicação de relação de agregação de objetos através de tipos de relacionamentos -- Uma relação de agregação indica que um objeto representa uma abstração de vários outros que o compõem. É um caso particular de dependência existencial, no sentido de que os objetos subordinados têm a sua existência dependente da ocorrência do objeto mais abstrato, e não tem sentido por si só. No entanto, além de indicar a dependência existencial, uma relação de agregação indica também transitividade. Assim, uma relação de agregação é mais intensa do que a dependência existencial, no sentido que um objeto que representa uma agregação não pode ser retirado de uma base de dados se ele estiver associado através de relação de agregação com algum outro. Um exemplo de um tipo de relacionamento que apresenta uma relação de agregação seria por exemplo uma DOCUMENTAÇÃO ser CONSTITUÍDA por LIVROS e/ou MICRO-FICHAS. Note-se que uma determinada MICRO-FICHA não terá sentido para uma base de dados de projeto de sistemas digitais se não fizer parte de uma DOCUMENTAÇÃO representada na Base de Dados.

Deve ser notado que uma relação de agregação ocorre quando deve ser expresso o fato de um objeto ser composto de outros, e além disso, que os objetos subordinados dependem da existência do objeto mais abstrato. Relacionamentos do tipo COMPOSTO DE entre

PLACA-DE-CI e CIRCUITO-INTEGRADO não representa um exemplo de uma agregação, pois um CIRCUITO-INTEGRADO não tem a sua existência dependente de alguma PLACA-DE-CI que o use.

IV.5.2.2. - Informações sobre Atributos.

As informações sobre atributos sempre se referem aos tipos de objetos e/ou de relacionamentos a que os tipos de atributos se associam. São aqui caracterizadas como informações sobre atributos porque são idênticas, não importando o fato de a associação ser feita com objetos ou com relacionamentos.

-- indicação de se um tipo de atributo pode ter apenas um ou vários valores para cada tipo de objeto ou de relacionamento --
Os atributos associados a objetos ou a relacionamentos, sejam eles de qualquer característica, podem ter apenas um valor, ou vários valores. Por exemplo a propriedade CUSTO de um PROJETO pode ter apenas um valor, ao passo que a Indicação de Tempo DATA-DE-ENTREGA-DE-RELATÓRIO desse projeto poderá ter várias ocorrências. Note-se que um mesmo tipo de atributo, associado a diferentes tipos de objetos ou relacionamentos poderá ter vários valores para um tipo e apenas um para outros.

Quando um atributo pode ter vários valores, ele é dito Multi-valorizado, e nesse caso múltiplos pedidos consecutivos de operações de colocar valor irão resultar na coleção de todos os valores colocados. Quando um atributo pode ter apenas um valor ele é dito mono-valorizado, e cada pedido consecutivo de opera-

ções de colocar valores irão substituindo os valores anteriormente colocados.

Aos atributos multivalorizados pode ser indicado um número máximo de ocorrências de valores que podem ser colocados. Ultrapassado esse número, passa a ser proibida a colocação de novos valores, até que algum valor seja retirado.

-- Indicação de obrigatoriedade da existência de um tipo de atributo -- Existem muitos atributos que podem ser colocados ou não em um objeto ou relacionamento existente na Base de Dados. No entanto existem tipos de atributos que devem obrigatoriamente ser colocados em determinados tipos de objetos ou de relacionamentos. Nesse caso, deve-se indicar para cada tipo de relacionamento ou de objeto que atributos de determinados tipos devem ocorrer, pois caso contrário a Base de Dados não pode considerar completa a informação disponível sobre cada objeto ou relacionamento daquele tipo.

Se um tipo de atributo é multi-valorizado e obrigatório, então o número mínimo de ocorrências deverá ser indicado também.

IV.5.2.3. - Informações sobre Associações entre Objetos e Relacionamentos e/ou Atributos.

-- Indicação da cardinalidade mínima e máxima de todos os tipos de relacionamentos -- Para cada tipo de relacionamento a que cada tipo de objeto possa estar associado, deve ser indicado quantos relacionamentos desse tipo podem no máximo ser associados, e

quantos relacionamentos desse tipo devem ser efetivamente associados no mínimo.

-- Indicação de exclusão mútua entre tipos de relacionamentos ou atributos, ou entre tipo de relacionamento e de atributo --
Sintaticamente, pode-se estabelecer que objetos de um determinado tipo podem ter relacionamentos de dois tipos associados, porém semanticamente podem ocorrer casos em que somente um dos relacionamentos possa estar associado simultaneamente, ou seja, qualquer um dos dois tipos de relacionamentos pode estar associado a uma instância de objeto daquele tipo, porém não pode existir nenhuma instância de objeto daquele tipo que esteja associado ao mesmo tempo com relacionamentos dos dois tipos.

Pode ocorrer que a exclusão não seja entre dois tipos de relacionamentos, mas sim entre um tipo de relacionamento e um tipo de atributo, ou entre dois tipos de atributos que ocorram em um determinado tipo de objeto. Nos três casos, a indicação de exclusão mútua é feita da mesma maneira.

A informação semântica de exclusão mútua entre entidades associadas a um mesmo objeto pode ser muito mais sofisticada do que a mera comparação entre dois tipos, tal como por exemplo, um determinado tipo de atributo não poder ocorrer se dois tipos de relacionamentos existirem simultaneamente em objetos de um determinado tipo. Porém, o MRD não se ocupa desses casos mais elaborados, ficando então a cargo dos programas de aplicação qualquer restrição que envolva mais do que dois tipos de relacionamento ou de atributo.

-- Indicação da distribuição de particularidades no conjunto de

objetos que compõem relacionamentos de equivalência -- Nos agrupamentos de objetos que constituem uma relação de equivalência, pode ocorrer que cada objeto envolvido, ou conjuntos de objetos envolvidos tenham particularidades que os distingua dos demais objetos do agrupamento. Por exemplo, tomando-se o relacionamento CONECTADO COM envolvendo objetos do tipo PINO dos CIRCUITO-INTEGRADOS que compõem uma PLACA-DE-CI, pode-se dizer que esse relacionamento estabelece uma relação de equivalência entre o agrupamento de pinos que estão inter-CONECTADOS. Em cada relação de equivalência, alguns PINOs apresentarão a particularidade de ser supridouros (OUTPUT), outros de ser entrada (INPUT) e outros, em alguns tipos de circuito, que podem ser as duas coisas (THREE-STATE).

Para que as particularidades sejam indicadas, é feita a indicação de que objetos devem, e quais objetos podem: ter determinados tipos de relacionamentos com que tipo de objeto destino; ou ter determinados tipos de atributos e/ou valores de atributos. Por exemplo, em um relacionamento de equivalência com o tipo de relacionamento CONECTADO COM associado a objetos do tipo PINO, apenas um objeto pode ter o atributo OUTPUT, e se houver ao menos um com o atributo THREE-STATE, então nenhum pode ter o atributo OUTPUT, porém deve existir ao menos um PINO com o atributo OUTPUT ou THREE-STATE.

-- Indicação da posição de cada relacionamento binário em cada relacionamento múltiplo -- Os relacionamentos múltiplos são compostos de relacionamentos binários, e pode ocorrer que um determinado tipo de relacionamento binário ocorra repetidamente. Além disso, é possível que os objetos destinos de cada tipo de

relacionamento binário que tenha ocorrência repetida possam ser de mais de um tipo. Torna-se assim necessário que se estabeleçam para cada relacionamento múltiplo a cardinalidade (mínima e máxima) de cada relacionamento binário que possa ocorrer, bem como os tipos de objeto destino de cada relacionamento binário que podem ou devem ocorrer.

-- Indicação da distribuição de particularidades no conjunto de objetos que compõem um relacionamento múltiplo -- Nos relacionamentos múltiplos que admitem a repetição de ao menos um tipo de relacionamento binário componente, pode ocorrer que cada objeto destino, ou conjuntos de objetos destino, dos relacionamentos binários repetidos, tenham particularidades que os distingua dos demais objetos destino de relacionamentos daquele tipo, particularidades essas expressas através de tipos de relacionamentos, tipos de atributos ou valores de atributos que os objetos destinos devem ter. Por exemplo, tomando-se o relacionamento múltiplo PRODUZ/EMPREGANDO para indicar as PLACAS-DE-CI que são PRODUZIDAS EMPREGANDO-SE determinadas PESSOAs, pode-se exigir que uma das várias PESSOAs tenha como valor do atributo tipo QUALIFICAÇÃO o valor PROJETISTA.

Para que as particularidades sejam indicadas, é feita a indicação de que objetos devem, e quais objetos podem: ter determinados tipos de relacionamentos com que tipo de objeto destino; ter determinados tipos de atributos; ter determinados tipos de atributos com determinados valores de atributos.

-- Indicação de implicação de existência e cardinalidade de tipos de relacionamentos ou tipos de atributos dependendo da existência

de outros tipos de relacionamentos ou tipos de atributos em objetos de cada tipo -- Existem casos em que um determinado tipo de relacionamento ou de atributo não é obrigatório em objetos de um determinado tipo, porém torna-se obrigatório em determinadas instâncias de objetos daquele tipo devido à ocorrência de outros tipos de atributos ou de relacionamento. No MRD, a implicação de existência de um relacionamento ou atributo de um determinado tipo, devido à ocorrência de um determinado tipo de relacionamento ou de atributo pode ser indicada.

Da mesma maneira, a indicação de implicação de existência de um entre dois tipos de relacionamento ou de atributo pode ser indicada, ou seja, situações em que por exemplo dois relacionamentos não são a priori obrigatórios, mas que obriga a que ao menos um deles deva existir.

A indicação de implicação de existência é feita especificando-se para cada tipo de objeto, os tipos de relacionamentos e de atributos que:

- existindo implicam na existência de outros tipos de relacionamento ou de atributo;
- não existindo implicam na existência de outros tipos de relacionamentos ou atributos.

IV.5.2.4. - Informações sobre Associações entre Atributos e Relacionamentos.

-- Indicação de exclusão mútua entre tipos de atributos de relacionamento em cada tipo de relacionamento -- Sintaticamente,

pode-se estabelecer que os relacionamentos de um determinado tipo podem ter atributos de dois tipos associados, porém semanticamente podem ocorrer casos em que somente um dos atributos possa ocorrer simultaneamente, ou seja, qualquer um dos dois tipos de atributos pode estar associado a um determinado relacionamento, porém não pode existir nenhuma ocorrência de relacionamento daquele tipo que esteja associado ao mesmo tempo com atributos dos dois tipos.

A indicação de exclusão mútua entre os tipos de atributos associados a cada tipo de relacionamento é feita indicando-se para cada tipo de relacionamento os pares de tipos de atributos que são mutuamente exclusivos.

-- Indicação de implicação de existência de tipos de atributos dependendo da existência de outros tipos de atributos em cada tipo de relacionamento -- Existem casos em que um determinado tipo de atributo não é obrigatório em relacionamentos de um determinado tipo, porém torna-se obrigatório em determinados relacionamentos daquele tipo devido à ocorrência de outros tipos de atributos no relacionamento. No MRD, pode ser indicada a implicação de existência de um atributo de um determinado tipo, devido à ocorrência de um atributo de outro tipo.

Da mesma maneira, a indicação de implicação de existência de um entre dois tipos de atributo em um determinado tipo de relacionamento pode ser indicada.

A indicação de implicação de existência é feita especificando-se para cada tipo de relacionamento os pares de tipos de atributos que:

- a existência de um tipo implica na existência do outro;
- a não existência de um tipo implica na existência do outro;

MQ-F
TQ-I
IV.5.2.5. - Uma Consideração sobre Ordem entre os Objetos Destino de Relacionamentos.

Os objetos destino de relacionamentos binários que ocorrem múltiplas vezes em um mesmo objeto origem podem ser ordenados segundo alguma forma de identificação que possuam (nome principal ou algum sinônimo). Além disso, como cada relacionamento é colocado separadamente, existe a ordem de colocação dos relacionamentos. O MRD possui operações para a obtenção de dados de relacionamentos em sequência, porém a sequência é sempre aquela da ordem em que os relacionamentos são colocados. A alteração dessa ordem, e a recuperação dos objetos destino de um relacionamento em ordem diferente da ordem de colocação é feita apenas através de seleção de todos os objetos que são destino de um determinado relacionamento e a posterior ordenação dessa seleção de objetos segundo o nome principal ou tipo de sinônimo de interesse. Assim, no MRD não existe nenhuma informação semântica explícita sobre a ordem de recuperação de dados, além da ordem de colocação dessas informações. A ordem é especificada explicitamente pelo programa de aplicação no instante em que a recuperação de um conjunto de informações é solicitado.

IV.5.3. - Especificação Semântica Avançada.

Os conceitos apresentados até este ponto devem poder ser estendidos, para que maiores recursos semânticos possam ser explorados por um SGBD apoiado no MRD. Toda a informação semântica apresentada até este ponto concentra-se na atribuição de um tipo para cada um dos objetos e dos relacionamentos da base de dados. No entanto, é freqüente que um tipo possa ser composto de outros, ou um determinado objeto sirva apenas para designar o tipo a outro.

A escolha dos conjuntos de tipos de objetos e de relacionamentos de uma aplicação deve ser feita de maneira a suprir todas as necessidades de representação do mundo real, ao mesmo tempo que permita a manipulação da descrição com facilidade e flexibilidade. Esses dois objetivos para a escolha dos conjuntos de tipos de objetos e relacionamentos são freqüentemente antagônicos e um compromisso deve ser encontrado.

Para permitir a escolha desses conjuntos de tipos, que representem todas as facetas de interesse do mundo real, e permitir que a complexidade do modelo não degrade o acesso e os algoritmos de manipulação desses dados, o MRD admite a existência de dois níveis de semântica. Pela própria existência dessa meta-classificação, as informações semânticas de um modelo são divididas em pelo menos três níveis:

- aquela que corresponde ao que o modelo trata como informação semântica básica;
- aquela que corresponde ao que o modelo trata como informação semântica avançada;
- aquela que permite a interligação dos elementos dos dois

níveis anteriores.

Para a modelagem da semântica básica, um conjunto de tipos de objetos e de relacionamentos é escolhido, definem-se suas tabelas léxicas, sua sintaxe e sua semântica. Os conjuntos de tipos devem ser escolhidos de maneira a que atendam a linha de tratamento de dados mais usada, mais significativa do modelo.

Refinando o modelo com informações correspondentes à semântica avançada, os tipos de objetos e de relacionamentos podem por sua vez ser constituídos de subtipos, e esses por sua vez também podem ser constituídos por outros subtipos. Por outro lado, alguns tipos podem ser agrupados com outros, ou com outros subtipos, e constituir supertipos.

Consideremos inicialmente os subtipos de objetos. Para efeito de facilitar esta discussão, será usado o termo "subtipo" significando "um objeto de um dado subtipo", uma vez que a definição do conteúdo da Meta-Base de Dados especifica a estrutura da Base de Dados.

Um subtipo indica que um objeto, além de ter o subtipo, é também do tipo básico do qual o subtipo faz parte. Assim, o conjunto de todos os objetos de um dado subtipo é um subconjunto do conjunto de todos os objetos daquele tipo. Um mesmo objeto pode fazer parte de mais de um desses conjuntos, visto que o conjunto dos subtipos não estabelece uma partição no conjunto dos objetos de um determinado tipo.

Um subtipo é especificado através da indicação de:

- um tipo, ou subtipo já previamente especificado, que será denominado Tipo Mestre do Subtipo;

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

O predicado que define um subtipo consiste em uma expressão booleana, tendo como operadores os conectivos E, OU e NÃO, e tendo como variáveis: a verificação dos valores de determinado atributo de cada objeto segundo regras de comparação adequadas aos valores próprios da característica daquele atributo; e a existência de determinado relacionamento, podendo ou não ser levados em conta atributos desse relacionamento ou o tipo do objeto destino.

Por exemplo, os objetos do tipo TIPO-DE-CI podem ter como atributos a dissipação de potência, o fato de serem analógicos ou digitais, serem comerciais ou restritos a uso próprio, etc. O tipo TIPO-DE-CI pode então ter subtipos definidos sobre esses atributos, tais como:

CI-DE-BAIXA-POTENCIA cujo predicado indicaria o atributo
{DISSIPACAO-DE-POTENCIA < 100}

ANALOGICO-DE-ALTA-POTENCIA cujo predicado seria
{CLASSE=analógico E DISSIPACAO-DE-POTENCIA >1000}

DIGITAL-COMERCIAL cujo predicado indicaria
{DISPONIBILIDADE=comercial E CLASSE=digital}

Note-se que um mesmo objeto pode ter mais de um subtipo. Dessa forma, os subtipos estabelecem uma hierarquia não estrita entre tipos e subtipos, no sentido de que não existem ciclos de subtipos, mas um mesmo subtipo pode ter mais de um tipo mestre. Por exemplo o objeto SN74LS00 do tipo TIPO-DE-CI seria também do subtipo CI-DE-BAIXA-POTENCIA e do subtipo DIGITAL-COMERCIAL.

Todos os tipos de atributos que qualquer um de seus subtipos puderem ter, um tipo também poderá ter. Porém nem todos os tipos de atributos de seu tipo mestre será válido em determinado subtipo. Por exemplo, como TIPO-DE-CI genérico, os objetos desse tipo poderão ter o atributo NUMERO-DE-PORTAS. No entanto os objetos do subtipo ANALOGICO-DE-ALTA-POTENCIA não deverão ter esse atributo.

O mesmo vale para os tipos de relacionamentos, que poderão ser restritos para determinados subtipos. Por exemplo, os objetos do subtipo CI-SEM-DOCUMENTAÇÃO, definidos pelo predicado (não tem relacionamento DOCUMENTADO POR), não poderá ter também o relacionamento PODE-SER-SUBSTITUIDO POR.

Deve ser ressaltado que o fato de um tipo mestre admitir sintaticamente todos os tipos de atributos e de relacionamentos que quaisquer um de seus subtipos admitem, podem trazer ambiguidades a uma base de dados. Por exemplo, a propriedade especifica o TIPO-DE-CI do subtipo AMPLIFICADOR-OPERACIONAL, tal como GANHO-EM-MALHA-ABERTA terá que ser propriedade de TIPO-DE-CI em geral, porém não terá nenhum sentido se for atribuído a um TIPO-DE-CI do subtipo DIGITAL-COMERCIAL.

Esse "relaxamento sintático" é necessário devido ao fato de que os objetos que efetivamente são armazenados são os que compõem o tipo principal, definido até o nível da semântica básica.

No entanto, as ambiguidades que poderiam ocorrer devido a isso podem ser evitadas obrigando que os tipos de atributos e de relacionamentos que se referem a subtipos apenas sejam manipulados pelas rotinas que tratam da semântica avançada, tanto para inclusão quanto para modificação e retirada, sendo proibidos o uso destas operações para as funções com nível semântico inferior.

Da mesma maneira que os objetos, os relacionamentos também podem ter subtipos, e que também são caracterizados por atributos dos seus relacionamentos mestres. Por exemplo, uma PESSOA pode participar de um PROJETO de diversas maneiras diferentes, e o relacionamento PARTICIPA poderia então ter o atributo MANEIRA. Os valores desse atributo poderiam ser "Chefe", "Técnico", "Consultor", etc. Poder-se-iam então definir os subtipos do relacionamento PARTICIPA como sendo CHEFIA, E'-TECNICO-DO, REALIZA-CONSULTORIA, etc.

Tanto nos subtipos de objetos quanto de relacionamentos, se o atributo usado no predicado de definição for único e Monovalorizado e com um domínio de valores pré-definidos e finito, então o conjunto de subtipos definidos constituirão uma partição no conjunto de objetos (ou relacionamentos) do tipo mestre.

Da mesma maneira que existem subtipos de objetos, é possível o desenvolvimento do conceito de supertipos de objetos, os quais corresponderão a composição de mais de um tipo ou subtipos diversos dando origem a novos objetos, que se constituem em objetos que representam conjuntos de objetos dos tipos e subtipos que compõem aquele supertipo. Deve-se ressaltar que não existem

supertipos de relacionamentos. Os elementos de um supertipo são os conjuntos de objetos dos tipos e subtipos que compõem o supertipo. Por exemplo, pode-se escolher alguns subtipos do tipo TIPO-DE-CI para constituírem o supertipo FAMILIA-LOGICA, sendo que cada elemento desse conjunto é um conjunto de TIPOS-DE-CI que constituem uma FAMILIA-LOGICA.

Todos os objetos que pertençam a qualquer um dos conjuntos de objetos que são elementos de um supertipo devem necessariamente ser de tipos que pertençam a uma única classe de objetos. Dessa forma, no esquema exemplo da Figura 14 podem ser estabelecidos supertipos que contenham conjuntos, aos quais pertencem simultaneamente por exemplo objetos do tipo ATIVIDADE e PLACA-DE-CI, mas não poderá existir supertipos cujos conjuntos componentes contenham simultaneamente objetos do tipo ATIVIDADE e RECURSOS, pois esses dois últimos tipos de objetos constituem classes distintas.

Os supertipos podem ter relacionamentos com tipos e subtipos, bem como atributos. Esses elementos associados a um supertipo caracterizam relacionamentos ou atributos que são identificáveis para todos os objetos de um dado supertipo, ou que caracterizam o supertipo como um todo. Por exemplo, poderia ser definido o relacionamento DOCUMENTADO EM (oposto ao relacionamento DOCUMENTA) entre um objeto DOCUMENTAÇÃO e um objeto FAMILIA-LOGICA, indicando que todos os objetos TIPO-DE-CI daquela FAMILIA-LOGICA seriam documentados no objeto DOCUMENTAÇÃO que DOCUMENTA aquele objeto FAMILIA-LOGICA.

Também os atributos de um supertipo são atributos que, se existirem nos objetos que compõem cada um dos conjuntos de tipos ou subtipos que compõem o supertipo, também são associados aos

objetos daquele conjunto. Por exemplo, todos os elementos de FAMILIA-LOGICA são constituídos de conjuntos de TIPO-DE-CI. Os objetos do tipo TIPO-DE-CI tem o atributo TENSÃO-MAXIMA. Os objetos FAMILIA-LOGICA podem ter os atributos TENSÃO-MAXIMA e NÚMERO-DE-CI-NA-FAMILIA. Como o atributo NÚMERO-DE-CI-NA-FAMILIA não é um atributo de TIPO-DE-CI, esse atributo apenas caracteriza a família como um todo, e não cada objeto que compõe uma FAMILIA-LOGICA. Por outro lado, o atributo TENSÃO-MAXIMA, sendo atributo tanto de FAMILIA-DE-CI quanto de TIPO-DE-CI, o valor definido pela FAMILIA-DE-CI será herdado por todos os TIPO-DE-CI daquela FAMILIA-LOGICA.

Os atributos de um supertipo, que não são herdados pelos objetos que constituem os conjuntos que compõem o supertipo, são obtidos a partir do conjunto de todos os objetos que constituem cada tipo que compõem o supertipo. Assim, cada um desses atributos deve ser definido através de uma operação especificada no domínio do tipo da característica daquele atributo, como por exemplo somatório, mínimo, máximo, média, etc. de uma propriedade com tipo de dado numérico.

Vale ressaltar aqui que apesar dos termos "Supertipo", "Tipo" e "Subtipo" sugerirem a associação de cada um deles com três níveis de uma hierarquia, isso não ocorre realmente, pois cada um deles referem-se a um conceito relativamente independente dos demais. O Tipo caracteriza objetos que efetivamente são armazenados na base de dados, podendo assim ser assumidos como básicos em qualquer Base de Dados. O conceito de Subtipo estabelece sozinho uma rede hierárquica não estrita, onde os tipos aparecem em algum nível, que não é necessariamente o mais sintético, uma vez que podem existir subtipos que são resultado da

união de tipos com subtipos. O conceito de supertipo por sua vez não tem nada a ver com a hierarquia que os subtipos estabelecem, correspondendo sim ao conceito de que um conjunto de tipos e subtipos passam a ser as instâncias de objetos de um supertipo.

Deve ser notado que, ao contrário de um subtipo, um supertipo caracteriza um novo "tipo" de objeto na Meta-Base de Dados. Assim, não existe nenhum objeto na descrição do subtipo DIGITAL-COMERCIAL, mas apenas do tipo TIPO-DE-CI com os atributos correspondentes, porque um objeto de um subtipo é um objeto do mesmo tipo que seu tipo mestre. No entanto, um objeto do supertipo FAMILIA-LOGICA não é um objeto de nenhum dos tipos de seus componentes, pois seus componentes se caracterizam por conjuntos de tipos e subtipos. São portanto objetos com existência própria, efetivamente armazenados na base de dados, os quais associam-se aos objetos que constituem os conjuntos de que são compostos por para-relacionamentos não explicitados em uma base de dados, mas sim explicitados na Meta-Base de Dados. Os para-relacionamentos traduzem-se na "hereditariedade" das informações do supertipo para os objetos dos tipos e subtipos que compõem os conjuntos que constituem o supertipo.

A existência de para-relacionamentos é caracterizada pelas informações semânticas na Meta-Base de Dados que definem implicitamente um relacionamento na Base de Dados. Dessa forma o para-relacionamento existe logicamente, definido pela Meta-base de dados, mas não é efetivamente armazenado na base de dados.

Deve ser ressaltado que os conceitos de supertipos e classes de objetos são independentes entre si. O conceito de classe diz respeito ao agrupamento de objetos que estão logicamente caracterizados pelas informações do objeto que restringe aquela classe,

tal como é o caso de uma dada ATIVIDADE, que somente é caracterizada dentro de um PROJETO.

Já o conceito de supertipo envolve os conjuntos de objetos que têm propriedades semelhantes e distinguíveis dentro de uma classe de objetos, independentemente da existência de objetos que forneçam um meio de caracterizar (ou identificar) cada objeto individualmente. Assim, enquanto a associação entre um objeto de determinado supertipo e um objeto que pertence a um conjunto que compõe aquele supertipo é efetuada através de um para-relacionamento que não é explicitado na base de dados, a associação entre um objeto qualquer e um objeto constrito pelo primeiro é explicitamente armazenada.

A introdução dos conceitos de subtipo e supertipo no MRD provoca o surgimento do problema de Equivalência Semântica, no sentido que dois conjuntos de todos os objetos de um determinado subtipo ou tipo podem ser constituídos sempre dos mesmos objetos. Esse fenômeno é indesejado, e vem sendo tratado teoricamente, tal como em [JAJ_83]. Esse problema é ainda mais importante no MRD quando se leva em conta que esse modelo permite a implementação e a aplicação prática de todos esses conceitos simultaneamente. A ocorrência de uma Equivalência Semântica não percebida em um esquema de Base de Dados empregando esse modelo pode levar a problemas de inconsistência na base de dados e perda do significado armazenado. O MRD oferece boas perspectivas de estudos do problema, quando permite a modelagem de situações reais em um ambiente de base de dados representando formalmente todos os elementos que levam à equivalência semântica, mas como modelo em

si não pode resolver esse problema, que deve ser levado em conta e evitado quando da realização de um esquema de dados.

IV.5.3.1. - As Rotinas de Manipulação da Base de Dados Usando a Semântica Avançada

As informações do esquema de dados que constituem semântica avançada são manipuladas pelo SGBD através de um conjunto de rotinas chamadas pelos programas de aplicação, da mesma maneira que as rotinas da semântica básica e da semântica fundamental. Uma parte desse conjunto contém rotinas que executam funções semelhantes às demais, chamadas de Rotinas de Manipulação da Base de Dados usando a Semântica Avançada, tais como colocar na Base de Dados objetos com determinados subtipos ou supertipos, e uma parte contém rotinas que efetuam testes na base de dados, chamadas Rotinas de Verificação da Base de Dados.

As Rotinas de Manipulação usando a Semântica Avançada podem colocar, modificar, posicionar, obter dados e retirar dados da base de dados, com a diferença que sempre estarão levando em conta que os objetos e relacionamentos afetados podem ter associações com subtipos e supertipos, executando todas as funções paralelas concernentes a cada ação tomada.

Por exemplo, as rotinas de manipulação de objetos usando a semântica avançada constituirão um conjunto derivado das rotinas de manipulação de objetos básicas apresentadas na seção IV.4.3, sendo que para cada uma das rotinas básicas deve haver uma correspondente empregando semântica avançada, porém que leva em

consideração toda a informação semântica disponível. Assim existirão as três rotinas de colocação de objetos, que são colocar objeto dado seu nome e tipo: na classe corrente; na classe indicada e; na classe corrente colocando também os relacionamentos e atributos automáticos.

Porém, usando uma rotina de colocar objetos que emprega a semântica avançada, pode ser especificado além do nome, o tipo, ou o subtipo ou o supertipo. Qualquer um deles que seja indicado, se o objeto colocado pertence a um conjunto de objetos que constituem um supertipo já definido, os atributos do objeto do supertipo correspondentes são verificados para avaliar se o novo objeto causa alteração em qualquer deles. Além disso, se for especificado o subtipo, e não o tipo do objeto, os atributos e relacionamentos que definem o subtipo são verificados, e colocados no novo objeto também. Por exemplo, se for solicitado para colocação na classe corrente o objeto SN74LS00 do subtipo DIGITAL-COMERCIAL, o atributo DISPONIBILIDADE é colocado com o valor "comercial" e o atributo CLASSE é colocado com o valor "digital".

Uma dificuldade ocorre quando um valor de atributo não é único para identificar o subtipo. Por exemplo, se o atributo usado no predicado de definição do subtipo for comparado por outro operador de comparação que não o "=", ou se o mesmo atributo for usado mais de uma vez, então, a operação não é bijetora e o inverso não pode ser determinado. Isso ocorre por exemplo com o subtipo ANALOGICO-DE-ALTA-POTENCIA, cujo atributo DISSIPACAO-DE-POTENCIA não pode ser identificado. Pode ocorrer que exista um valor normalmente aceito como típico ("default"). Nesses casos, ao definir-se um subtipo, no conjunto de valores de atributos

padrão deve ser especificado um valor de atributo pré-estabelecido que será colocado quando o objeto é criado.

Outros trabalhos têm procurado evitar que ocorra esse tipo de ambiguidade, sugerindo que a especialização de um tipo em subtipos somente possa ocorrer quando o conjunto de subtipos é finito e conhecido a priori, com os valores (tipicamente de atributos) que caracterizam qual o subtipo de cada instância do tipo sendo escolhidos de um conjunto discreto. É por exemplo o caso do trabalho, hoje clássico de Smith e Smith [SMI_77b], que destina-se ao modelo relacional, e caracteriza subtipos como abstrações de generalização, onde os vários itens genéricos são agrupados em relações específicas definidas quando a Base de Dados é inicializada. No entanto, para evitar restrições desnecessárias, o MRO pode admitir subtipos definidos através de predicados que usa relações de ordem, e o fato de valores típicos poderem ser definidos já estende a sua aplicabilidade ao menos nos casos em que isso é possível.

Deve ser notado aqui que o fato de um objeto ser de um determinado subtipo não é explícito, mas sim obtido do tipo mestre - que esse sim é explicitado para o objeto - e do predicado de definição do subtipo. Assim, se um objeto for criado com a indicação de um determinado subtipo, e em operações posteriores um atributo ou relacionamento que caracteriza esse subtipo for modificado, o objeto perde ou muda seu subtipo, enquanto o seu tipo somente pode ser alterado através de uma operação explícita.

Isso pode não ser sempre desejável, pelo menos do ponto de vista semântico, embora existam casos em que isso seja desejável. Por exemplo, se a propriedade que caracteriza um certo TIPO-DE-CI

ser do subtipo LINEAR-DE-ALTA-POTENCIA for alterado, aquele TIPO-DE-CI poderá ter mudado seu subtipo, e isso não tem significado nenhum, uma vez que não é exequível que um certo TIPO-DE-CI mude dessa forma. Por outro lado, se o atributo de um objeto do tipo PESSOA do subtipo TECNICO-DE-LABORATORIO for alterado para indicar que aquela PESSOA é do subtipo PROJETISTA, essa alteração terá correspondido a uma mudança do mundo real que é válida.

É necessário que exista um dado semântico adicional no modelo para que essa distinção entre a exequibilidade ou não de alteração de um atributo ou relacionamento que esteja envolvido em um predicado indicando o fato de um objeto ser de um subtipo possa ser feita. A princípio, essa distinção pode ser feita através de um significado semântico básico ao qual todos os atributos ou relacionamentos poderiam estar associados, qual seja, a possibilidade de mudança de valor de um tipo de atributo, ou mudança de um objeto destino de relacionamento. Porém isso nem sempre é satisfatório, uma vez que esse significado pode não ser associado ao tipo de atributo ou de relacionamento em si, mas sim a cada caso particular. No exemplo acima, o mesmo atributo que poderia mudar o subtipo de uma PESSOA de PROJETISTA para ESTAGIARIO não teria sentido, uma vez que corresponderia a fazer a PESSOA deixar de ter atributos que não podem ser retirados.

Para evitar tais inconsistências, o MRO não assume a escolha de uma opção, deixando-a a cargo do programa de aplicação, o qual faz a escolha através do conjunto de rotinas que usar para efetuar a alteração. Usando as rotinas que verificam somente a semântica fundamental, quaisquer alterações são possíveis, enquanto que usando as rotinas que verificam a semântica avançada, as mudanças que causam alteração de um subtipo são

impedidas.

Para as demais operações de manipulação da base de dados, considerações semelhantes se aplicam, e devem ser levadas em conta. As operações de posicionamento e obtenção de dados não alteram a base de dados, mesmo que possam detectar inconsistências. Por exemplo, a operação de posicionar no primeiro objeto de um determinado subtipo, seguida de operações de obter o código do objeto e avançar podem determinar que o atributo NÚMERO-DE-CI-NA-FAMILIA do objeto de supertipo associado ao subtipo indicado está inconsistente, porém a inconsistência não é corrigida.

Também deve ser notado que uma operação envolvendo a semântica avançada que altera o conteúdo da Base de Dados, solicitada por um programa de aplicação, pode ocasionar a execução de operações de outros tipos. Por exemplo, colocar ou retirar um objeto de um determinado tipo pode originar a necessidade de modificar atributos de um objeto de um supertipo que está associado ao tipo do objeto colocado ou retirado.

As rotinas de Verificação da Base de Dados são rotinas que verificam aspectos específicos da base de dados, porém da mesma maneira que as rotinas de consulta da base de dados não causam alterações na base de dados. O objetivo dessas rotinas é confirmar a consistência de toda a semântica conhecida da base de dados. A execução dessas rotinas leva em conta tanto a semântica expressa no esquema de dados da aplicação quanto a definição explícita de padrões de testes, especificados através de Autômatos na Meta-Base de dados [TRA_82].

Deve ser lembrado que a avaliação de uma base de dados por uma rotina de teste é feita através da indicação para aquela

rotina de cada um dos objetos da Base de Dados. Uma verificação completa da base de dados deve ser feita por um programa de aplicação, que solicita seguidamente a execução daquela determinada rotina, para cada um dos objetos que possam estar inconsistentes, possivelmente contidos em uma seleção de objetos.

O fato de que a verificação de cada objeto deve ser solicitada explicitamente deve-se à grande quantidade de processamento normalmente envolvido para a determinação do resultado do teste, e do correspondente tempo e recursos computacionais exigidos. Através da solicitação explícita, o usuário pode determinar quando a avaliação deve ser feita. Além disso, é freqüente que a base permaneça inconsistente durante alguns estágios, mas que a própria aplicação que tenha levado à inconsistência a faça deliberadamente e restaure a consistência no final de uma seqüência de operações.

A existência de três níveis de rotinas, executando as mesmas operações é a causa de uma base de dados poder ficar inconsistente, porém é necessário a existência desses três níveis para um aumento de eficiência da operação do SGBD. A existência de rotinas de verificação da base de dados permitem que se garanta a consistência quando essa consistência se torna necessária.

IV.6. - O Uso de uma Base de Dados MRO em Ambiente Multiusuário.

Para que uma base de dados baseada no Modelo de Representação de Objetos possa ser usada para acesso multiusuário, é necessário que se incorporem ao SGBD recursos

para apoiar o acesso concorrente à Base de Dados. Conforme foi descrito no capítulo III, o MRO admite naturalmente o esquema de proteção de acesso através de travamento por predicado, usando para isso o conceito de classes de objetos.

Dessa forma, cada vez que uma aplicação necessita dos dados de um determinado objeto, a classe de objetos que contém esse objeto é travada para acesso por aquela aplicação. Podem ser previstos a princípio dois tipos de travamento:

- para leitura, que admite que mais de uma aplicação possa ter acesso simultâneo à mesma classe para leitura;
- para escrita, que exige que somente uma aplicação possa ter acesso à classe.

Em operação normal, qualquer usuário tem inicialmente acesso para leitura na classe global, e indicando sucessivamente vários objetos para consideração, várias classes vão se tornando acessíveis. Usando esse esquema para controlar a proteção de acesso, sempre que um objeto for indicado para consideração, deve ser também indicado se a classe que o objeto considerado abre deve ser aberta para apenas leitura ou para leitura e escrita.

O uso de uma base de dados apoiada no MRO, através de um Sistema de Gerenciamento de Dados para apoio ao desenvolvimento de projetos ou produção de engenharia, é naturalmente feito com cada aplicativo reportando-se a dados específicos que naturalmente dizem respeito a poucas classes. Além disso, as classes que são acessadas por um aplicativo e por um usuário em geral não são as mesmas que são acessadas por outro aplicativo ou outro usuário. Dessa forma, a probabilidade que uma mesma classe seja necessária para mais de uma transação simultaneamente fica bastan

te baixa.

Considere-se o desenvolvimento de um determinado aspecto de um projeto. Isso normalmente é efetuado por um projetista usando algum tipo de editor interativo, que lhe permite ir colocando novos dados, alterando os já existentes, até que o projetista esteja satisfeito, e aquele aspecto do projeto é considerado terminado. O simples fato de aquele aspecto do projeto requerer atenção especial por parte de um projetista caracteriza que esse aspecto deve ser um Item de Configuração de Projeto (ICP), que deve ter atenção por parte do Gerenciador de Configuração de Projeto. Esse aspecto irá então constituir um ICP, e poderá ou não envolver outros ICPs.

Sendo um ICP, aquele determinado aspecto teve que ser identificado como tal, e lhe foi atribuído identificação e um registro de estado para acompanhamento, o qual indica continuamente seu estado para acesso. A identificação corresponde no MRD à existência de um objeto, que restringe a classe de objetos que o descrevem. Para que pudesse ser acessado pelo projetista, seu registro de estado deveria estar indicando que ele estava em desenvolvimento, e a partir do instante em que o projetista passasse à edição, indicando aquele objeto para consideração, seu registro de estado indicaria o fato, evitando que outros projetistas pudessem também acessá-lo.

Os objetos constritos pelo objeto considerado que identifica o ICP, com todos os relacionamentos entre eles, constituem o próprio ICP, tal como ele é projetado, e como tal passível de ser submetido a todos os testes de análise e aceitação para o Gerenciador de Configuração de Projeto. Entre os objetos que constituem o próprio ICP podem existir outros ICPs. Para efeito

dos testes, e para o rastreamento de modificações e criação de novas versões, é necessário que os ICPs que são componentes de cada ICP sejam explicitados como tal. Assim o objeto que identifica um ICP deve estar relacionado diretamente por todos os outros ICPs que o compõem, através de um relacionamento de tipo COMPOSTO POR.

Quando da definição do ICP, o Gerenciador de Configuração, além da identificação como um ICP, teria solicitado também a especificação dos seus requisitos técnicos. Essa especificação corresponde à indicação de todos os objetos constritos àquele ICP que podem se relacionar com objetos não constritos por ele, ou seja, objetos fora da classe de objetos considerada por aquele objeto, e que especificam o contexto onde o ICP é usado. Essa indicação pode ser feita relacionando-se o objeto que identifica o ICP com todos os objetos que especificam seu contexto, através de relacionamentos do tipo CONTEXTO.

Dessa forma, as três funções básicas de identificação de um item de configuração de projeto correspondem diretamente à:

- **definição de ICP:** indicação de um objeto como capaz de restringir uma classe de objetos, e especificação dos atributos que descrevem e parametrizam essa capacidade;
- **identificação de requisitos técnicos do ICP:** indicação de todos os objetos constritos pelo objeto que identifica o ICP, que se relacionam com objetos não constritos por ele, através de relacionamentos do tipo CONTEXTO;
- **atribuição de identificadores a ICPs e a seus componentes:** indicação de todos os demais ICPs que compõem aquele ICP através de relacionamentos do tipo COMPOSTO POR.

Para a obtenção de todos os relacionamentos que associam um objeto constrito por um objeto que identifica um ICP e um objeto externo à classe constrita pelo ICP, deve-se então verificar os relacionamentos dos objetos que fazem parte do CONTEXTO daquele ICP. Na Base de dados MRO, essa especificação corresponde ao conjunto de todos os relacionamentos existentes entre qualquer dos objetos que compõem a classe constrita pelo objeto identificador do ICP, e objetos externos a essa classe.

Dessa forma, as duas únicas formas de acesso às informações referentes a cada ICP correspondem ao objeto que identifica essa ICP, e aos relacionamentos que interligam os objetos constritos por aquele objeto com objetos fora dessa classe de objetos. Para fazer-se o controle de acesso às classes, devem ser controladas essas duas formas de acesso.

O acesso através da identificação da ICP é a mais fácil, uma vez que é feita explicitamente, e permite qualquer tipo de controle que se pretenda implantar.

O acesso por seus relacionamentos já requer um procedimento mais sofisticado. A princípio, pode-se considerar a implementação de um teste efetuado em qualquer operação de acesso a um objeto da base por parte de um aplicativo, para verificar se o objeto que constringe o objeto acessado está disponível para aquele aplicativo. Em sistemas multiusuário com a Base de Dados centralizada, esse enfoque é suficiente, e por si só permite um controle de acesso adequado. Porém, se o ambiente de processamento for um sistema distribuído, ou contiver estações de trabalho inteligentes que podem manipular parte da base de dados armazenada na própria estação de trabalho, então esse procedimento não é suficiente.

Em um sistema multiusuário distribuído, a base de dados pode estar "espalhada" em vários processadores, e em alguns casos, com cópias de partes da base em mais de um processador. Consideremos que apenas uma dessas cópias é "oficial", e que as demais são cópias de trabalho, feitas para acesso local durante um tempo limitado, e que após algum tempo serão eliminadas, ou integradas na Base oficial. Dessa forma, o Gerenciamento de Configuração deve controlar quando as cópias são feitas, manter um registro de todas as cópias que existem num dado momento, e controlar quando e como uma integração de uma cópia com a base oficial é feita.

Para efeito dos aplicativos que manipulam a base de dados, mesmo que exista mais de uma cópia ou parte da base oficial em um dado nodo da rede, apenas uma cópia é reconhecida de cada vez, justamente a parte que aquele aplicativo deve acessar.

Os objetos que estiverem presentes em uma cópia terão junto de si todas as informações correspondentes a esse objeto, envolvendo todos os atributos e todos os relacionamentos associados a esses objetos. Uma cópia contém o conjunto de todos os objetos de um ou mais ICPs, de forma que contém conjuntos inteiros de dados que são controlados por uma operação de travamento ou liberação de dados.

Para efeito da discussão que se segue, é considerado que os objetos dos quais existem cópias na parte da base de dados acessada por um aplicativo são "objetos locais", e os que não existirem nessa parte da base de dados são considerados "objetos remotos".

Assim, é necessário que, enquanto o processamento está sendo feito sobre os dados específicos de um (ou de um conjunto limitado de) ICP, então todos os relacionamentos sejam feitos exclusi-

vamente com objetos desse ICP, que num sistema distribuído são locais a cada aplicativo. Isso pode ser feito facilmente usando-se identificadores locais para designar objetos remotos. Como as classes não são efetivamente compostas de objetos, mas sim de identificadores de objetos, pode-se criar um identificador local para um objeto pertencente a uma classe remota, e fazer temporariamente a ligação de cada um daqueles relacionamentos com objetos identificados dessa forma.

O que é feito na realidade, é a criação de um "objeto fictício" na parte local da base de dados para cada objeto remoto que contenha um relacionamento com um objeto local. Cada um dos objetos fictícios mantém uma informação de qual é o objeto real a que ele corresponde, e a identificação local que é dada a esse objeto. O que caracteriza esse objeto como um objeto fictício, é que além dessa identificação, ele mantém a identificação que deve ser usada para acessar o objeto real quando for efetuada a integração da cópia local com a base de dados oficial.

Quando se faz a abertura de uma classe, se essa abertura corresponder a processamento externo ao computador que mantém essa classe na base de dados, efetua-se uma cópia de todos os objetos, relacionamentos e atributos dessa classe, sendo que para cada um dos objetos externos a essa classe que se relacionam com objetos dela, cria-se na cópia criada um objeto fictício, que mantém informação de qual é o objeto real a que aquele corresponde.

Durante a manipulação dessa cópia, qualquer novo relacionamento que envolva objetos externos origina a criação de novos objetos fictícios, e o comportamento do sistema atuando como estação de trabalho é efetuado como se a cópia que ele manipula fosse toda a base de dados.

Quando uma tarefa de manipulação é dada por encerrada, essa parte da base de dados - considerada pelo Gerenciador de Configuração como uma ICP - é integrada na Base de Dados "oficial", e todos os objetos fictícios são eliminados, convertendo-se todos os relacionamentos que com eles existiam diretamente para o objeto real correspondente.

A discussão acima mostra que dois níveis de proteção são necessários. Um corresponde à verificação de permissão de acesso sempre que ocorrem operações sobre classes, tais como abertura fechamento e acesso a objetos, e outro corresponde às operações de cópia e reunião de partes da base de dados, operações essas correspondentes a classes e ICPs sobre as quais operações de manipulação, congelamento e descongelamento são solicitadas.

Para facilitar a execução das operações de cópia e integração de partes da base de dados, a estrutura física de acesso aos dados deve prever essas operações. Isso pode ser conseguido juntando as informações correspondentes a cada objeto nos mesmos registros físicos, evitando que um mesmo registro físico armazene informações sobre mais de um objeto, incluindo-se aí os atributos e os relacionamentos desses objetos. Dessa forma, quando é efetuada a cópia, basta que sejam copiados fisicamente esses registros, e fazer uma tabela de acesso aos registros como se fossem páginas virtuais. Assim, os códigos de acesso internos das informações podem ser feitos com referência a essa tabela de acesso virtual, tornando desnecessária qualquer reformatação da base de dados.

IV.7. - O Uso do Modelo de Representação de Objetos em uma Base de Dados para Engenharia.

No capítulo II deste trabalho foram identificadas as necessidades que um Sistema de Gerenciamento de Dados para Engenharia deveria ser capaz de suprir para as aplicações de sistemas de apoio ao projeto, desenvolvimento e produção de sistemas de engenharia. Tendo sido apresentado o Modelo de Representação de Objetos, construído com o intuito de satisfazer aquelas necessidades, é conveniente verificar agora como cada uma delas pode ser satisfeita.

(n1) - O sistema deverá receber e fornecer os dados de cada aplicação em formato adequado a cada uma em particular.

O MRD trata cada item de informação individualmente, e reconhece as associações que existem entre cada item de maneira específica para cada objeto que se coloca em apreço. Dessa forma, a troca de dados entre o SGBD e a aplicação é feita através de cada item individualmente. Para que o programa de aplicação receba algum dado do SGBD, é necessário que ele solicite cada item que o compõe explicitamente, explicitação essa que é feita no instante em que o pedido é solicitado. Assim, os dados são fornecidos exatamente da forma esperada pela aplicação. O mesmo ocorre quando dados são passados para o SGBD.

(n2) - A Base de Dados deve admitir acesso concorrente e pode ser distribuída.

O acesso concorrente é implementado em um SGBD através de

uma estrutura de controle de acesso que divide os dados de uma Base de Dados em unidades de acesso denominadas blocos, e permite ou impede o acesso solicitado por uma aplicação a cada bloco. O MRO estabelece um mecanismo lógico intrínseco de se estabelecer os blocos de acesso, que corresponde às classes de objetos, o qual além de permitir o controle de acesso, faz com que os blocos bloqueados por um programa de aplicação correspondam exatamente aos dados necessários e suficientes para a execução de um programa, minimizando os problemas de espera de liberações e conflito de acesso. O mesmo mecanismo pode ser usado para estabelecer o critério de partição dos dados para a alocação distribuída da base de dados. Além disso, conforme discutido na seção IV.6, o conceito de Classes de Objetos juntamente com o conceito de Gerenciamento de Configuração pode ser empregado para estabelecer critérios para a replicação de informações em diferentes nodos de um sistema.

(n3) - O Esquema de Dados de uma aplicação deve poder mudar mesmo depois que a Base de Dados seja inicializada, através da existência de um Meta-esquema de dados.

O MRO tem a descrição do esquema de dados intrinsecamente representado em uma Meta-Base de Dados, e isso a priori permite que o esquema de dados seja alterado mesmo depois que a Base de Dados já tenha sido inicializada. No entanto, dependendo do tipo de alteração que é realizada na Meta-Base de Dados, podem ser necessárias alterações na Base de Dados. Assim, operações de definição de novos tipos de objetos, relacionamentos e atributos, bem como a inclusão de novas

associações sintáticas entre entidades da Meta-Base de Dados, não causam nenhuma alteração na Base de Dados. No entanto a retirada de uma dessas informações da Meta-Base de Dados origina a necessidade de se eliminar ou corrigir da Base de Dados todos os dados envolvidos com o Meta-Dado eliminado. Além disso, a inclusão de uma informação semântica adicional na Meta-Base de Dados ocasiona a necessidade de se verificar todos os dados envolvidos na Base de Dados, quanto ao cumprimento da nova regra semântica incluída.

(n4) - Deve ser possível armazenar grande quantidade de Meta-Dados.

Em um programa de aplicação, os dados sobre os dados armazenados, comumente denominados Meta-dados são constituídos de informações semânticas sobre os dados que podem ser reconhecidas e expressas pelo SGBD ou pelo próprio programa de aplicação. Assim, os Meta-Dados de uma aplicação são em parte representados no Esquema da Aplicação, e em parte representados pela própria implementação do programa de aplicação. Usando o MRD, uma parcela bastante grande dos Meta-Dados podem ser representados na Meta-Base de dados. Porém, como informação semântica sempre pode ser adicionada - não é possível a especificação integral da semântica, tal como é possível a especificação integral da sintaxe - deve ser considerado que sempre haverá informação semântica que não é armazenada na Meta-Base de Dados. Uma característica de Meta-dados é ter uma estrutura altamente heterogênea, o que dificulta, ou mesmo impede, a sua representação em Bases de Dados tradicionais. Uma Base de Dados usando o MRD por outro

lado aceita a especificação de dados a cada objeto de forma individual, não estruturada, independente de como é a estrutura de qualquer outro objeto da Base de Dados. Isso permite que um programa de aplicação passe para a Base de Dados quanta informação semântica for necessária ou conveniente, além daquelas já representadas na Meta-Base de Dados.

(n5) - Manutenção de várias versões de um mesmo projeto.

Cada aspecto de um projeto que deve ser tratado através de várias versões terá no MRD um objeto que o representa. Os detalhes de cada aspecto são no MRD também tratados através de objetos, com seus respectivos relacionamentos e atributos associados. Sendo necessário estabelecer várias versões de um mesmo objeto, corresponde no MRD a se definir um objeto para cada versão do aspecto que é multiplicado pelo número de versões. Os detalhes de cada versão daquele aspecto correspondem cada qual aos objetos que são constrictos pelo objeto correspondente a cada uma das versões desse aspecto. As informações de um projeto que são idênticas nas várias versões correspondem a objetos que não detalham o aspecto do projeto para o qual existem versões, e dessa forma são representadas por objetos que não são constrictos pelos objetos que definem cada versão, não sendo dessa forma repetidos. Os relacionamentos que existem associando os objetos que variam com cada versão, com os objetos que não variam são efetuados naturalmente através dos relacionamentos inter-classes, comuns ao MRD. Além disso, os para-relacionamentos indicados pela existência de Supertipos (frequentemente usados para o agrupamento de informações que especificam os

requisitos necessários a um aspecto, independentemente de uma particular versão) podem propagar para cada versão as informações globais que são especificadas para indicar as restrições de projeto com que qualquer versão deve se ater, permitindo que objetos constritos a uma versão herdem as especificações de projetos, e possam refiná-las.

(n6) - A identificação de dados, ou blocos de dados deve poder ser feita tanto explicitamente, como através da forma como alguns dados em particular se associam.

Usando o MRD, a identificação explícita de um objeto é feita através de um nome, ou através de qualquer sinônimo desse objeto. Relacionamentos entre objetos também podem estar associados a atributos, permitindo que dados não muito complexos possam ser identificados através do relacionamento entre os dados em particular. Além disso, um relacionamento pode também identificar um objeto, e dessa forma, estabelecendo-se qual é o relacionamento que envolve dados em particular, pode-se obter acesso a objetos que descrevem esse relacionamento através de qualquer grau de complexidade.

(n7) - Múltiplos projetos devem poder compartilhar a mesma base, e usar dados em comum.

Usando o conceito de classes, o MRD provê uma separação tanto lógica quanto física dos objetos e seus dados associados, constritos por uma classe. Todos os dados de um projeto podem dessa forma estar constritos ao objeto que identifica aquele projeto em particular. Os dados genéricos, usados por qualquer projeto podem fazer parte da classe global, ou de

outras classes abertas em paralelo com cada classe de projeto. Associada a uma implementação de Base de Dados que ofereça capacidade de processamento concorrente, ou mesmo distribuído, o MRO permite que vários projetos sejam desenvolvidos concorrentemente, partilhando a mesma base de dados e os dados em comum.

(n8) - Devem ser suportados muitos tipos de dados.

O MRO permite que muitos tipos de dados, de diferentes estruturas possam ser atendidos. Normalmente, os Sistemas de Gerenciamento de Dados existentes atendem aos vários tipos de dados necessários fornecendo unidades elementares (campos ou atributos) de dados, aos quais é dado significado pelo programa de aplicação. O MRO permite que as estruturas de dados lógicas mais utilizadas em uma aplicação possam ser atendidas pelo próprio Sistema de Gerenciamento de Dados, através das várias características de atributos que podem ser reconhecidas.

(n9) - São necessários recursos para o atendimento de aplicações gráficas.

Os dados gráficos são tratados pelo MRO essencialmente através de atributos com característica de gráficos. Normalmente o que ocorre é a existência de um objeto na base de dados que representa um item de interesse do mundo real. Esse item pode ter uma existência própria, além do fato de ter dimensões e outras características espaciais. Modelado por um objeto, existirão então atributos do objeto que representam as propriedades, sinônimos, regras associadas ao

objeto, e também informações gráficas. Dependendo de quais serão os dados manipulados pelos atributos com característica de gráficos numa implementação, esses atributos podem descrever todas as características geométricas de qualquer item do mundo real. Note-se que informações gráficas indicando uma particular representação gráfica (tais como uma janela em uma estação gráfica) não são intrínsecas ao item em si, e portanto não devem fazer parte dos atributos do objeto que o representa, mas sim de algum outro objeto da base de dados, que poderia representar então os dados relativos às formas de representação de interesse em cada estação de trabalho disponível.

(n10) - Atender a usuários em variado grau de experiência e conhecimento de Sistemas de Bases de Dados.

Sendo usado em um Sistema de Bases de Dados, a facilidade que o Modelo de Representação de Objetos pode apresentar para ser usado depende bastante da particular implementação do SGBD. No entanto, sendo um modelo de alto conteúdo semântico, a proximidade de representação dos dados com os dados do mundo real, conhecidos dos usuários, é naturalmente maior do que a oferecida com os modelos tradicionais de Bases de Dados, muito mais voltados às características de máquina do que o MRD. Como exemplo disso, a possibilidade de se expressar informações não somente de forma direta através de objetos e seus tipos básicos, mas também através de subtipos e supertipos, permite que um usuário possa se comunicar de forma altamente significativa com o SGBD em vários níveis de abstração, enquanto o próprio SGBD pode "deslocar" as

informações para outros níveis através de "herança" ou "generalização" de dados.

IV.8. - Conclusões

Nesse capítulo foi apresentada uma estrutura de dados, que tem por objetivo a implementação de um SGBD que manipula um alto conteúdo semântico associado aos dados da base de dados. Um conjunto de conceitos semânticos bastante extenso foi também apresentado, juntamente com um conjunto de rotinas que poderiam administrar os dados em um ambiente semântico intenso e constituir o SGBD.

Os conceitos semânticos apresentados não são novos, uma vez que vêm sendo estudados tanto no âmbito matemático, quanto no âmbito de Sistemas de Gerenciamento de Informações. Porém, a integração de todos esses conceitos em um único modelo, bem como a proposta de uma estrutura unificada que os suporte não foi apresentada antes.

Além disso, a estrutura proposta admite a implementação de uma estrutura física que mantém o conteúdo semântico expresso no esquema, sem que se tenha de recorrer a estruturas de registros para a implementação de um modelo que é essencialmente semântico, o que é também diferente dos trabalhos que têm sido propostos.

Os Sistemas de Gerenciamento de Bases de Dados que existem comercialmente são todos baseados em estruturas de registros. Esse enfoque tem sido seguido, porque os Modelos de Rede, Hierárquico e Relacional são intrínsecos e logicamente modelos apoiados

em estruturas de registros.

O Modelo Entidade-Relacionamento é o primeiro, dentre os mais divulgados, a ser baseado em semântica. No entanto, a implementação de SGBDs apoiados nesse modelo é preconizada como sendo feita através de relações, ou seja, através de estruturas de registros.

O maior conteúdo semântico do ME-R tem sido explorado através da modelagem de bases de dados em Diagramas de Entidades e Relacionamentos (DE-R), e depois adaptadas para uma modelagem que pode ser aceita por SGBDs apoiados nos outros modelos. O MRO ao contrário, incorpora em um DE-R muitas outras informações semânticas, e que a seguir são representadas pelo SGBD sem nenhuma adaptação maior do que a mera representação do modelo representado graficamente para a representação de uma Linguagem de Definição de Dados do próprio SGBD.

Quanto ao nível semântico do SGBD, duas considerações devem ser levantadas.

A primeira é que, uma vez definidas matematicamente, outros conceitos podem ser incorporados ao modelo, uma vez que ele não é constituído de um conjunto hermético de conceitos, mas admite que novas funções possam ser incorporadas. Não se quer dizer com isso que qualquer novo conceito possa ser incorporado, porém de qualquer maneira o MRO admite um espaço bem maior para isso do que os demais modelos existentes.

A segunda é que os conceitos matemáticos representados pelo MRO são muito mais abrangentes do que os envolvidos pelos outros modelos. Por exemplo, o Modelo Relacional apresenta operações e estrutura próprios para representar os conceitos da álgebra das relações. O MRO envolve desde a Álgebra de Boole, com as opera-

ções de verificação de regras, com os operadores lógicos agindo sobre elementos distintos, passa pela teoria dos conjuntos, através dos conjuntos de objetos que satisfazem àquelas regras, e atinge também a álgebra das relações quando esses conjuntos de objetos são considerados com todos os seus atributos e relacionamentos.

Assim, a maior abrangência de conceitos respalda a afirmação de que o MRO modela uma correspondentemente maior quantidade de casos do mundo real.

As situações que interessam diretamente ao modelo são as que ocorrem em ambientes de desenvolvimento, projeto e produção de bens nas variadas áreas de atividade. Nesse sentido, um levantamento das necessidades para um Sistema de Gerenciamento de Dados para Engenharia foram comparadas com maneiras como o MRO se adequa para satisfazê-las.

Sem dúvida, para outras aplicações, outros modelos podem ser mais indicados que o MRO, tais como em aplicações em que exista a necessidade de recuperação de registros através dos valores de

campos que no MRO seriam preferencialmente representados através de propriedades, ou em aplicações em que a recuperação de registros seqüencialmente é prioritária, casos em que um SGBD apoiado no MRO apresentaria uma eficiência menor que os sistemas apoiados em modelos tradicionais. Porém, essas situações são raras em aplicações de projeto e produção de engenharia, e o MRO é nitidamente concebido para situações opostas a essas. De qualquer forma, são situações que, embora bem realizadas pelos SGBDs apoiados em modelos tradicionais, são trabalhosas para uma realização manual, reforçando a característica de que o Modelo de Representação de Objetos é voltado para a representação o mais próximo possível da lógica com que os projetistas desenvolvem seus projetos.

C A P Í T U L O V

MAQUINAS ESPECIAIS PARA BASES DE DADOS

V.1. - Introdução.

Com o advento e consolidação dos Sistemas de Gerenciamento de Bases de Dados(SGBDs), um volume muito grande de informações passou a ser colocado em computadores, gerando Bases de Dados de porte muito grande. Originalmente, todos os sistemas de gerenciamento desenvolvidos eram executados em computadores convencionais, usando a tradicional arquitetura de Von Newmann. Porém, com a proliferação desses sistemas, e com o aumento de seu porte, passou a ser cogitada a possibilidade de se desenvolver computadores dedicados a essa aplicação, com arquitetura adaptada à execução dos processos envolvidos.

As primeiras propostas nesse sentido começaram a surgir no início dos anos 70, sendo comumente aceito ter sido Slotnick o primeiro a introduzir o conceito de uma arquitetura dedicada à execução dos SGBDs, em [SLO_70]. Em pouco tempo, um consenso da necessidade e das especificações de uma Máquina de Base de Dados tomou forma, porém a maneira como as especificações poderiam ser atingidas constituiu e constitui objeto de muito estudo, sendo

hoje possível distinguir três fases pelas quais o seu desenvolvimento já passou.

A primeira fase foi dominada pela concepção de máquinas comumente denominadas "Discos Associativos" [SLO_70] [COU_72], em que a característica principal consistia em se incorporar lógica adicional nas cabeças de leitura de dispositivos de armazenagem de cabeça fixa, tornando a procura inicial de dados da Base de Dados realizável a nível das próprias cabeças de leitura dos dispositivos, somente fazendo-se a transmissão de dados das trilhas ou setores em que alguma chave procurada fosse localizada. Essa fase durou aproximadamente toda a década passada, e serviu para a caracterização e consolidação dos conceitos de Máquinas de Base de Dados. Um desses conceitos é o de se encarar a Máquina de Base de Dados como um Dispositivo de Retaguarda ("Back End Machines"), capaz de atender um ou vários computadores que executam os programas de aplicação [CAN_74].

Na segunda fase dominou o conceito de agrupar-se um conjunto de processadores e distribuir o processamento entre esses processadores, sendo a especificidade da máquina como um todo dedicada ao processamento de SGBDs pela elaboração criteriosa dos algoritmos que distribuíssem os processos executados pelos vários processadores. Nessa fase também foram sugeridas arquiteturas que englobavam lógica especial para tarefas específicas, tais como ordenadores em hardware, e processadores usando arquitetura SIMD (múltiplas unidades lógico-aritméticas controladas por apenas uma unidade de controle, processando então com uma única instrução múltiplos dados (Single Instruction, Multiple Data)).

Vários projetos em desenvolvimento hoje ainda podem ser considerados como pertencentes a essa fase, porém um novo

conceito vem tomando forma, e que poderia ser usado para caracterizar a terceira fase. Esse é o conceito de que a própria estrutura da base de dados deve ser adaptada para tirar proveito de qualquer arquitetura dedicada ao processamento do SGBD. Nesse sentido, não apenas a organização física dos dados em dispositivos de armazenagem deve ser adaptado à arquitetura, mas também a sua estrutura lógica, que em alguns casos de propostas recentes chegam a propor alterações no próprio modelo de dados da Base de Dados.

Nas três seções seguintes serão apresentados brevemente alguns dos projetos existentes de Máquinas de Base de Dados, que são mais significativos em termos de contribuições para o desenvolvimento desses dispositivos. Vale lembrar, que muitos desses projetos não foram implementados efetivamente, sendo que muitas das conclusões a que se chegaram foram obtidas de simulações e estimativas de performance. A apresentação procura ser feita numa sequência progressiva dos conceitos que são incorporados nos projetos, o que de uma certa maneira implica numa ordem cronológica de concepção dos projetos, sem que seja esse no entanto, o critério da sequência de apresentação.

V.2. - Discos Associativos.

A grande maioria dos primeiros projetos de Máquinas de Base de Dados consistiam em se colocar um processador associado a cada cabeça de leitura de dispositivos de cabeça fixa (discos ou tambores), tal como mostra a Figura 20, em que existe uma cabeça para cada trilha gravada no meio de armazenagem. Esse conceito

foi usado para a proposta inicial de Slotnick em [SLO_70], e foi seguido em vários projetos subsequentes.

Figura 20 - Máquina baseada em processadores por trilha.

Na proposta de Slotnick, os processadores associados às cabeças de leitura de unidades de discos eram denominados processadores de trilha ("per-track processors"), e eram usados para facilitar a localização dos registros da Base de Dados que deveriam ser usados para as operações subsequentes.

Os processadores de trilha eram carregados com as informações de quais dados procurar, em que campo de dados dos registros da base de dados. A procura então era encetada em paralelo em todas as trilhas do dispositivo, fazendo com que toda a base de dados fosse pesquisada em apenas uma revolução do disco. Pela proposta original, nenhuma outra ação além de comparação era

realizada pelos processadores de trilha, caracterizando assim perfeitamente a denominação de Discos Associativos.

O processador que atuava como controlador do dispositivo era o mesmo que executava o programa de aplicação. Assim a operação do SGBD como um todo era executada da seguinte maneira:

- a operação a ser executada era analisada pelo processador controlador, que enviava aos processadores de trilha os dados que permitiam identificar o registro (ou os registros) que seria usado para aquela operação;

- os processadores de trilha, no espaço de uma revolução do disco seleccionavam esse (ou esses) registro, e apenas ele era enviado para o processador controlador;

- o processador controlador executava então a operação sobre o registro (ou registros) recebido, e prosseguia para a próxima operação, sendo para efeito dos processadores de trilha, a operação considerada encerrada.

A concepção aparentemente simples desse arranjo trazia no entanto consideráveis problemas. O primeiro deles correspondia ao tratamento de vários processadores de trilha tentando enviar registros ao processador controlador, ocasionando uma congestão do duto de dados que interligava o Disco Associativo com o computador controlador. Mesmo que esse duto pudesse ter capacidade de transferir todos os dados necessários, ainda assim o processador controlador teria problemas com a taxa de dados de todos os processadores de trilha enviando dados ao mesmo tempo.

A simplicidade com que se pretendia implementar os processadores de trilha esbarrava também na necessidade imposta pelo problema de sobrecarga do duto de dados, quando um mesmo proces-

sador poderia localizar em sua trilha, em uma mesma revolução do disco, vários registros contendo a chave de procura solicitada, fazendo então com que se necessitasse de um processador com capacidade local de armazenagem de dados correspondente à própria trilha. Essa solução corresponde a se ter no total das memórias locais a cada processador de trilha tanta memória quanto o próprio disco poderia armazenar, o que torna essa solução inexequível. Além disso, a existência de "buffers", para equilibrar a velocidade de acesso dos dispositivos de armazenagem com a velocidade do duto poderia trazer sérios problemas de segurança dos dados, devido a problemas de queda de energia ou cancelamento de operações já iniciadas [LAN_78].

Para evitar isso, a alternativa seria aceitar que, havendo congestionamento do duto, outras revoluções adicionais seriam usadas para permitir o envio ao processador controlador dos registros que não tivessem conseguido acesso ao duto na revolução (ou revoluções) anteriores.

Um projeto interessante pelo tratamento ao problema do congestionamento de muitos processadores de trilhas querendo transmitir dados simultaneamente foi dado por Lin e outros em [L55_76], onde foi proposta a máquina RARES. Nessa arquitetura, os bytes que compõem uma única tupla são distribuídos um em cada trilha, do mesmo setor. Assim, na operação de comparação dos dados lidos pelas cabeças de leitura com a chave de pesquisa, cada processador teria que comparar um byte, e ao final da leitura de cada setor de um byte, se todos os processadores concordassem quanto ao resultado da comparação, então uma tupla teria sido localizada e seria então enviada para

um "Buffer" de saída, que a transmitiria ao computador principal. Dessa forma, apesar de ser necessário uma taxa de transferência elevada (quando várias tuplas consecutivas fossem selecionadas), nunca ocorreria a necessidade de transmissão simultânea de mais de uma tupla ao mesmo tempo. O problema inerente a uma arquitetura desse tipo é que o tamanho de cada tupla fica limitado ao número de trilhas do dispositivo, e tuplas menores que esse tamanho causam um desperdício muito grande da capacidade do dispositivo.

Originalmente, a concepção de Discos Associativos não era destinada especificamente a qualquer modelo de dados em especial, porém a capacidade de processamento paralelo dos dispositivos tornava-os especialmente apropriados para a execução do modelo de dados relacional (tal como [L55_76] e [BAB_79]), e em breve todos os projetos concebidos passaram a levar em conta esse modelo como o objetivo principal (na maior parte dos casos até com exclusividade).

Um problema associado à arquitetura de Discos Associativos para a execução do modelo relacional é que, embora a operação de seleção de dados seja realizada com eficiência, qualquer outra operação que envolva mais de uma relação (união, projeção, seleção envolvendo a comparação de atributos de mais de uma relação, etc.) envolvia uma revolução do disco para a comparação de cada duas tuplas, o que degradava muito a execução dessa máquina.

Hawthorn e DeWitt em [HAW_82] estimaram que, para um processador de trilha ser capaz de processar todos os dados lidos por uma cabeça de leitura em um disco típico (cita-se para o levanta-

mento dos dados o Acionador Ampex 9200), cada processador de trilha deveria ser capaz de executar dois milhões de instruções por segundo, o que não representa em absoluto, ainda hoje, um processador simples de ser implementado, mesmo levando-se em conta o reduzido conjunto de instruções necessárias. Além disso, os autores mostraram que, comparando a execução de um SGBD usando o modelo relacional em um computador tradicional, com uma implementação usando discos associativos, a necessidade de revoluções inteiras para cada registro que devesse ser comparado com outro registro, levava a uma performance em que no melhor dos casos a implementação usando discos associativos era duas vezes mais rápida do que uma implementação bem feita em um computador tradicional. Conclui-se então que, dada a muito maior complexidade de um disco associativo em relação a um computador tradicional, a relação Custo/Benefício não justifica de maneira alguma o uso de discos associativos.

V.3. - Máquinas de Base de Dados que não Restringem a Estrutura Lógica dos dados.

A partir do final da década passada, vários projetos de Máquinas de Base de Dados passaram a tomar forma, sendo que alguns deles chegaram mesmo a ser implementados, e são usados até hoje tanto para a avaliação de novos algoritmos, quanto para o estudo de novas soluções. A seguir são apresentados os projetos que de uma forma geral não implicam na existência de dados adicionais na base de dados para a manutenção da própria base de dados.

Desenvolvida na Universidade da Flórida, a máquina CASSM (Context Addressed Segment Sequential Memory) [SSU_79] pode ser vista como uma evolução do conceito de Disco Associativo, pois apesar de manter a existência de lógica associada a trilhas do disco, admite a possibilidade de um processador poder atender a mais de uma trilha, ao mesmo tempo em que tem o seu conjunto de operações bastante aumentado. O conceito básico da máquina CASSM é que os dados são mantidos em segmentos de memória, que podem estar em uma (dispositivos de cabeça fixa) ou mais trilhas (dispositivos de cabeça móvel), porém uma dada trilha somente é tratada por um único processador. Os dados são mantidos em arquivos, os quais por sua vez são constituídos de registros hierarquicamente estruturados. A preocupação fundamental para o desenvolvimento dessa máquina, e da linguagem usada para controlar suas operações (CASDAL) foi atender ao modelo Hierárquico de dados, se bem que tanto o modelo relacional quanto o de rede foram analisados pelos autores [SSU_78].

Cada processador era implementado como um processador serial (tratando um bit de cada vez), em uma arquitetura usando conceitos de "Pipeline". Dessa forma, os dados lidos em uma cabeça de leitura eram tratados bit a bit inicialmente por um módulo que consistia em uma "memória FIFO", que ajustava o tempo de processamento com o tempo de leitura, e permitia ao próximo módulo a determinação de se uma operação em andamento já estava encerrada ou não. O segundo módulo tinha o encargo de avaliar se a operação sobre uma sequência de bits já estava encerrada ou não, e determinar a validade da sequência através de um esquema de bits de paridade. O próximo módulo estava encarregado da

localização de bits determinados como não usados, e fazer a compactação e recuperação de registros apagados. O módulo seguinte realizava a operação solicitada sobre os registros, quais sejam: seleção de registros; determinação de Mínimos e Máximos; comparação com registros já lidos e marcados para comparação; obtenção de totais. Ao módulo seguinte, denominado de módulo de seleção e interagindo ciclicamente com o anterior, estavam atribuídas as tarefas de guardar a chave de comparação e um possível registro lido anteriormente para efetuar as comparações entre registros. Por fim, existia um módulo que cuidava da restauração do esquema de paridade e da temporização de escrita, em uma cabeça distinta da cabeça de leitura. Todos os dados lidos eram invariavelmente reescritos por essa cabeça. A Figura 21 ilustra a constituição de um desses processadores.

Figura 21 - Estrutura de um Processador da máquina CASSM

O esquema geral da máquina CASSM permitia que os dados fossem livremente comunicados entre os vários processadores através de um duto interno, através do módulo de seleção. Para isso todos os processadores estavam sincronizados por um único sinal, e um ciclo da máquina CASSM correspondia a um período de

rotação de todo um segmento de dados, mais um tempo (denominado "gap time") onde ocorriam as transferências de instruções entre o processador controlador e os processadores de segmento, e onde as operações de agregação de dados aos arquivos eram efetuadas.

Uma característica importante da Máquina CASSM é que os dados eram armazenados por atributos, e não por registros. Assim, cada valor associado a um atributo era armazenado uma vez só, e os registros eram compostos por ponteiros para os dados reais.

A Máquina de Base de Dados DBC (DataBase Computer) [BAN_78] [BAN_79] foi desenvolvida na Universidade Estadual de Ohio em Columbus como um Dispositivo de Retaguarda para o processamento de dados, que pode ser acessada por um ou mais processadores onde são executados os programas de aplicação, e que do ponto de vista da máquina DBC são tratados coletivamente como o sistema de execução de programas (PES). Sua arquitetura é baseada em dois laços de processamento, que tem como intersecção um processador denominado "Processador de Comando e Controle da Base de Dados" (DBCCP), tal como mostrado na Figura 22.

O processador DBCCP recebe os pedidos de acesso à base de dados provenientes do PES, divididos estes em dois conjuntos: Comandos Preparatórios, que consistem de pedidos de informações e alterações da estrutura da Base de Dados (definição de campos e registros); e Comandos de Acesso, que requerem processamento sobre os dados da própria base. Além disso, esse processador controla a operação de todos os demais componentes, garantindo a consistência do processamento.

Os laços que compõem a máquina DBC são compostos por processadores e memória, por onde os dados passam em sequência, à

maneira de um "pipeline". O laço de dados, composto pelo subsistema de Memória de Massa (MM) e processador de segurança (SFP), recebe do DBCCP instruções para recuperar da MM as trilhas que contêm as informações solicitadas. Esses dados são então processados pelo SFP para garantir a permissão de acesso e a consistência dos dados recuperados, efetuando também a extração dos campos e registros necessários, e passando-os então ao DBCCP.

Figura 22 - Estrutura da Máquina de Base de Dados DBC.

O Laço de Estrutura é composto pela Unidade de Transformação de Palavras-Chaves (KXY), Memória de Estrutura (SM), Processador da informação da Memória de Estrutura (SMIP), e Unidade de Transformação de Índices (IXU). Os pedidos de acesso recebidos do PES pelo DBCCP são decodificados pelo KXY, para se obter os índices de acesso da definição dos registros e campos envolvidos.

Esses índices são passados para a SM, que atua como uma memória cache que mantém a informação de quais blocos da MM armazenam as informações de cada registro da base de dados. Usando esses índices, o SMIP realiza operações de união e/ou interseção com esses índices, para restringir o conjunto de blocos que deverão ser acessados. Por fim o IXU decodifica os índices de blocos em endereços físicos de acesso a trilha e setores do MM.

A Base de Dados é armazenada usando uma estrutura interna que procura tirar proveito do Laço de Estrutura, restringindo o número de acessos que são necessários para obter cada item de informação desejada, e essa estrutura reflete-se na forma como os pedidos devem ser solicitados pelo PES. Essa estrutura procura tirar partido de uma estrutura associativa de Hardware que pode localizar os endereços dos dados a partir de seus valores. Devido ao fato de não ter sido considerado exequível a implementação de memória associativa para armazenar toda a base de dados, ou toda a informação de estrutura da base de dados, foi desenvolvido o conceito de uma memória associativa por blocos ("Partitioned content-addressable memory" (PCAM)). Tanto a Memória de Massa MM quanto a Memória de Estrutura SM são constituídas de PCAMs.

Dessa forma, a estrutura da base de dados é própria à máquina DBC. Bases de Dados já existentes usando os modelos de rede, hierárquico ou relacional podem ser suportadas por essa máquina desde que submetidos a um processo de transformação. Porém devido à existência de uma interface de adaptação para os pedidos de acesso oriundos dos programas de aplicação já escritos usando esses outros modelos, os programas de aplicação em si não precisariam ser conceitualmente alterados.

A máquina DIRECT [DEW_79] tem sido desenvolvida na Universidade de Wisconsin em Madison. É descrita como um conjunto de processadores independentes associados a um subsistema de memória CCD, projetado especificamente para dar suporte a Sistemas de Gerenciamento de Bases de Dados Relacionais. Sua arquitetura é baseada em um conjunto de processadores que executam todas as operações de manipulação da base de dados, denominados Processadores de Consulta ("Query Processors"), e um Processador Controlador ("Back-End Controller") que recebe os pedidos de acesso do processador que executa os programas de aplicação, tal como é mostrado na Figura 23.

Figura 23 - Estrutura da Máquina de Base de Dados DIRECT.

Todos os dados manipulados por essa máquina são agrupados em páginas de tamanho fixo de 16K Bytes, que são tratadas como um todo. Cada relação da base de dados pode usar uma ou mais páginas, porém uma página estará sempre associada a apenas uma relação. Quando um pedido de acesso é efetuado por um programa de aplicação, esse mesmo processador compila o pedido em uma sequência de operadores da álgebra relacional, que é enviada para o Processador Controlador. Este aloca um ou mais Processadores de Consulta para cada operação que possa ser executada em paralelo, ao mesmo tempo que providencia que os dispositivos de armazenagem tornem disponíveis os dados necessários.

Os módulos do subsistema de memória CCD tem cada um capacidade para armazenar uma página de dados, e são constituídos por memória que mantém os dados continuamente circulando, simulando discos de alta velocidade. O conceito fundamental de operação dessa máquina é portanto ainda o emprego de memória associativa para realizar as operações de seleção e projeção de dados de uma base de dados usando o modelo relacional.

Dessa forma, as páginas podem ser transferidas diretamente entre os dispositivos de armazenagem e o Subsistema de Memória. Além disso, uma matriz de interconexão permite que qualquer dispositivo de armazenagem e qualquer processador de consulta possam se conectar a qualquer módulo do subsistema de memória, permitindo que qualquer número de processadores possam ler ao mesmo tempo um módulo de memória, permitindo o paralelismo de operações que utilizem uma mesma relação em mais de uma operação relacional.

Cada processador de consulta tem memória local capaz de armazenar uma página inteira de dados, e mais toda a informação

necessária para realizar as operações necessárias sobre uma página. Assim, um processador pode realizar as operações de seleção e projeção armazenando os dados de interesse à medida que estes vão "circulando" pela conexão estabelecida pela matriz de interconexão. Quando uma página fica completa, ou a operação é terminada, então a página é copiada num módulo do subsistema de memória e daí fica disponível para gravação em um dispositivo de armazenagem ou para uso de qualquer outro processador.

A máquina DIRECT foi implementada usando um processador PDP 11/45 como processador de execução dos programas, um processador PDP 11/40 como Processador Controlador, e processadores PDP 11/03 com 56 K Bytes de memória como Processadores de Consulta.

A máquina de base de dados desenvolvida na Universidade de Toronto, Canada, denominada RAP [OZK_77], destina-se a ser usada para executar SGBDs relacionais, e pode ser vista como uma extensão do princípio de associar processadores diretamente às cabeças de leitura de unidades de disco, porém envolvendo processadores mais poderosos, capazes de executar múltiplas operações sobre os dados a que têm acesso.

Fundamentalmente, RAP que é usado como um processador de retaguarda, consiste de três componentes principais tal como é mostrado na Figura 24:

- um processador controlador, que permite o acesso a partir do computador principal, que executa os programas de aplicação;
- uma unidade estatística;
- um conjunto de processadores denominados "células", os quais estão associados às cabeças dos dispositivos de disco.

Figura 24 - Estrutura da Máquina RAP.

O processador controlador controla a execução dos pedidos feitos pelo computador principal, a qual é realizada pelas células. A comunicação entre o computador principal e o Processador Controlador é efetuada por um duto paralelo de 16 bits, com um controlador de DMA que permite o acesso direto à memória desse processador, e além disso franqueia o acesso ao duto interno de controle e dados, permitindo que o computador principal possa escrever ou ler dados diretamente das memórias locais das células.

Existem três versões diferentes da máquina RAP, designadas por RAP.1, RAP.2 e RAP.3 [SCH_79]. A primeira versão usa efetivamente dispositivos de disco como a memória de dados das células, ao passo que as outras versões tiveram substituídos os discos por módulos de memória CCD, organizadas de maneira a simular "trilhas" onde os dados permanecem permanentemente "circulando", tal como na máquina RAPID.

O ponto alto da máquina RAP é que ela pressupõe a organização da estrutura da base de dados de maneira a facilitar a recuperação dos registros de interesse para uma dada operação. Isso é feito admitindo-se a existência de relações internas que fazem a correspondência entre as relações lógicas e as células que contêm os dados. Junto com os dados são mantidos campos de bits de marcação ("Mark Bits") que podem ser usados pelo programa de aplicação como uma área de trabalho para marcar determinados registros específicos dentro de uma relação, os quais são usados pelo processo de seleção associativa para testar os dados de interesse. Um bit em especial indica um registro apagado, o que o torna transparente para qualquer programa de aplicação.

A máquina SABRE ([VAL_84], [KAR_81]), em desenvolvimento na Universidade de Paris VI, é baseada na utilização de processadores simples, atuando como autômatas para a seleção de tuplas armazenadas em dispositivos de disco. Cada unidade de disco tem associado um desses processadores, denominados Processadores de Filtragem (Figura 25), os quais selecionam as tuplas baseados em predicados que especificam como uma tupla deve ser considerada ou não. As tuplas são enviadas para uma memória cache, onde se tornam disponíveis a um outro conjunto de processadores, denominados Processadores de União, que são capazes de realizar as operações de união e projeção sobre relações da base de dados.

A memória cache é acessada por esses processadores através de uma malha de interconexão de alta velocidade, e conhecendo o tipo de processo que é executado pelos processadores de União, é capaz de prever determinados tipos de solicitação que poderão ser feitos por aqueles, e antecipa a troca de informações entre seus

bancos de memória e os processadores de filtragem. Os processadores de união por sua vez dispõem de memória local para armazenagem das relações que resultam das operações de união e projeção, e transferem para a memória cache páginas inteiras de tuplas sempre que estas estão completas.

Figura 25 - Estrutura da Máquina de Base de Dados SABRE.

V.4. - Máquinas de Base de Dados que Especificam a Estrutura Lógica dos Dados.

Com a evolução de vários projetos de Máquinas de Base de Dados, surgiu a oportunidade de se analisar cada solução apresentada, realizar-se testes de performance de cada projeto, e efetuar avaliações de algoritmos que pudessem ser usados para executar as funções necessárias a um SGBD em uma arquitetura especial. Por isso, a partir do início desta década, vários estudos foram

efetuados, tanto para a avaliação de desempenho das máquinas propostas [SHU_84], quanto para o desenvolvimento de algoritmos especiais ([BOR_81], [BIT_83]) e de novas máquinas que aproveitassem as conclusões desses estudos.

V.4.1. - Técnicas de Análise de Algoritmos e de Projetos de Máquinas de Base de Dados

Um dos primeiros estudos a comparar as várias máquinas de base de dados existentes, e que estabeleceu um conjunto de testes padronizados que poderiam ser usados para comparar o desempenho dessas máquinas foi realizado por Hawthorn e DeWitt em [HAW_82]. Nesse estudo os autores propõem que se uniformize o tratamento dado às várias máquinas, estabelecendo-se diagramas que descrevam a máquina alvo através de:

- processadores de controle, que correspondem aos meios que a máquina utiliza para exercer as funções de controle dos demais componentes que executam as operações na base de dados;
- processadores celulares, que correspondem às unidades de processamento que efetivamente executam as operações na base de dados;
- controlador de unidades de disco, que efetuam todas as operações físicas de leitura e gravação de dados.

Dessa forma, cada um dos elementos que compõem uma máquina podem ser parametrizados de uma maneira também uniforme, e

algoritmos padronizados de simulação podem fornecer dados comparativos entre duas máquinas.

Para efetuar-se a simulação, os autores propõem a execução simulada de três tipos de consulta, que se destinam a avaliar o desempenho da máquina em um ambiente relacional, e que são os seguintes:

- Consultas curtas, onde apenas são solicitadas operações de seleção de um único conjunto de dados de uma relação;
- Consultas envolvendo múltiplas relações, com operações de projeção e união;
- Operações de agregação de dados, onde uma relação deve ser dividida em vários segmentos, com base nos valores de um atributo, sendo que a seguir uma função é aplicada a cada tupla de cada partição, sendo que o resultado é um valor para cada partição.

Os autores especificam ainda qual deve ser a base de dados, já preparada, que deve ser usada para a simulação, em termos de definição de tuplas e número de tuplas por relação. Nesse artigo são analisadas as máquinas baseadas em Discos Associativos, RAP, CASSM, DBC, DIRECT e CAFS, bem como todas elas são comparadas com a execução de uma implementação da linguagem relacional INGRES [STO_76] em um computador convencional. O resultado dessas comparações mostrou que embora as máquinas de base de dados examinadas fossem em geral pelo menos 3 vezes mais rápidas para a execução de consultas do primeiro tipo, nenhum projeto se revelou significativamente mais rápido para executar as consultas mais complexas.

Outros autores posteriormente seguiram o modelo adotado para efetuar análises comparativas e previsões de desempenho de outros projetos desenvolvidos posteriormente. Por exemplo, Hillyer e outros em [HIL_86] descrevem a arquitetura de uma máquina projetada por eles e a submetem aos mesmos critérios adotados por Hawthorn e DeWitt.

Figura 26 - Máquina de Base de Dados em Estrutura de Árvore Binária.

Essa máquina é baseada em um grande número de processadores de oito bits de pequena capacidade, cada um com uma reduzida memória local, conectados entre si de maneira a compor uma árvore binária. Tal como é mostrado na Figura 26, cada um dos processadores dos níveis mais altos de hierarquia dessa árvore estão conectados as unidades de disco do sistema, através de processadores de maior capacidade computacional, que efetuam a leitura desses dispositivos e passam a seqüência de dados lidos para os

seus subordinados, os quais, a critério da operação sendo executada continua passando cada seqüência recebida para seus respectivos subordinados. Controlando a execução de todos os processadores conectados a discos existe uma rede que providencia a distribuição dos dados e comandos para a execução de operações na base, bem como recolhe o resultado das consultas para serem enviados ao computador principal. Uma rede conectando os processadores de mais baixo nível permite a troca de informações necessária à execução de operações que envolvem os dados armazenados em mais de um disco.

Para a avaliação comparativa dessa máquina, os autores usaram um sistema com 8191 processadores, e obtiveram resultados de execução de 2 a 10 vezes melhores que as melhores arquiteturas analisadas em [HAW_82].

O emprego de arquiteturas usando múltiplos processadores apresenta intrinsecamente o problema de um alto tráfego de mensagens entre os dispositivos de armazenagem de dados e os processadores. Devido à natureza das operações que devem ser executadas em uma base de dados relacional, selecionando tuplas seqüencialmente em relações, e comparando seqüências de tuplas (para as operações de união e projeção), muitas máquinas propostas tinham uma arquitetura SIMD, onde esse problema poderia ter várias soluções. No entanto Boral e DeWitt em [BOR_81], mostraram que uma arquitetura de múltiplos processadores apresenta notável melhora de desempenho quando comparada com as arquiteturas SIMD, a despeito da maior complexidade necessária na rede de conexão entre os processadores e dispositivos de armazenagem de dados.

Para a solução desse problema os autores propõem três

alternativas:

- Aumentar o tamanho das páginas transferidas de cada vez.
Essa solução apresenta o inconveniente de diminuir o paralelismo possível na execução das consultas;
- Melhorar a malha de interconexões. Essa solução, embora sempre utilizada em maior ou menor grau, não pode ser usada como única alternativa, pois sua complexidade é crescente com o número de dispositivos interligados e com a eficiência que dela se espera.
- descentralizar o controle de execução das consultas, deixando-o ao encargo dos processadores que executam os programas de aplicação. Essa solução é apresentada pelos autores como a melhor alternativa, pois auxilia a distribuir os picos de utilização da malha de interconexão.

Nas máquinas de base de dados que têm sido propostas, os algoritmos usados para executar as operações estão intimamente ligados à própria arquitetura da máquina de base de dados. Dessa forma, estudos e análises comparativas de diferentes algoritmos alternativos para a implementação de uma dada operação tem se mostrado de muita importância. Bitton e outros em [BIT_83] propuseram uma organização modelo para representar arquiteturas de multiprocessadores, onde os algoritmos poderiam ser simulados. Esse modelo é composto por:

- um conjunto de processadores de propósito genérico;
- um conjunto de dispositivos de armazenagem de dados;
- uma malha de interconexões ligando os processadores aos dispositivos de armazenagem através de uma memória

cache de alta velocidade.

Os autores propõem nesse mesmo trabalho diferentes algoritmos para operações de ordenação, projeção e união de relações em arquiteturas de multiprocessadores, e fazem uma discussão de como esses algoritmos executariam na máquina DIRECT.

Usando esse mesmo modelo como base para discussão, Cesarini e outros em [CES_85] propõem uma notação gráfica para a representação de consultas a bases de dados usando o modelo relacional, apoiada na Teoria dos Grafos [HRR_69]. A seguir, usando conceitos dessa teoria desenvolvem um conjunto de regras que permitem obter uma transformação da consulta em uma árvore de execução de operações, que pode ser usada para analisar e descobrir paralelismos de operações que podem tirar proveito de uma arquitetura de multiprocessadores.

Qadah & Irani em [QAD_85a] estabelecem um critério para a classificação de Máquinas de Base de Dados operando sobre o modelo relacional, baseado em três características:

- a) o fato da arquitetura ser SISD (apenas um processador), SIMD (um processador baseado em uma única unidade de controle controlando várias unidades lógico-aritméticas) ou MIMD (vários processadores independentes).
- b) o local onde o processamento tem lugar: se no próprio disco, em memória dedicada, ou híbrido. No primeiro caso, cada unidade de disco dispõe de um processador que executa todas as operações diretamente nas trilhas dos discos envolvidos. No segundo, blocos de informações que possam

conter as informações relevantes são transferidas para um complexo memória/processador dedicado. No terceiro, algumas operações são realizadas diretamente no disco (tal como seleção de registros), e outras são realizadas através de cópias de partes do disco.

- c) o nível de indexação suportado: toda a base de dados, todas as relações ou páginas de informação. Essa característica diz respeito ao volume da base de dados que deve ser pesquisado para realizar as operações solicitadas. Máquinas da primeira categoria devem consultar todo o espaço de dados onde a base de dados está armazenada, as da segunda categoria podem restringir a pesquisa a algumas poucas relações, e as da terceira podem localizar em algum tipo de unidade de armazenagem (páginas) o local onde a pesquisa deva ser efetuada.

Além dessas três formas de classificação, uma quarta pode ser colocada, que corresponde ao fato da máquina enfatizar algum determinado modelo de dados. Assim, existem máquinas voltadas ao modelo relacional e máquinas que suportam igualmente qualquer um dos modelos de rede, hierárquico e relacional. A máquina proposta no capítulo VII, introduziria ainda uma terceira classe, que seria uma máquina voltada para um modelo não convencional, no caso, o Modelo de Representação de Objetos.

Após a apresentação da classificação acima, Qadah e Irani levantam considerações especulativas sobre a composição ideal de uma máquina de base de dados, chegando à conclusão que esta deve ser uma máquina MIMD, com o processamento sendo todo executado em

memória, e com capacidade de indexação a nível de páginas. Além disso, sustentam que a base de dados deve dispôr de mecanismos que permitam localizar rapidamente as páginas na memória secundária que possam conter as informações desejadas, de maneira a concentrar em uma pequena parte da base de dados a região onde os dados poderão ser encontrados.

Baseados nessa discussão, propõem então a máquina MIRDm (Michigan Relational Database Machine), voltada para a execução de SGBDs relacionais, e compostas por quatro conjuntos de processadores especializados em executar as operações de seleção de dados da base de dados, controlar a rede de comunicação de dados entre os processadores, atuar como memória cache para os processadores de execução de operações, e processadores de controle e interfaceamento com os computadores que executam os programas de aplicação. Uma descrição mais detalhada da operação de união de relações, da forma como pode ser realizada nessa máquina é descrita em [QAD_85b], mostrando com maiores detalhes o funcionamento da máquina MIRDm, juntamente com uma comparação de propostas alternativas dadas por outros autores, tais como [BIT_83], [VAL_84], [DEW_79] e [BAB_79].

Modelos especiais que possam aproveitar melhor as características de arquiteturas usando múltiplos processadores também têm sido estudados. Kerr e outros em [DEM_85] propõem o Modelo de Dados Baseado em Atributos (Attribute-based Data Model), o qual é essencialmente o modelo relacional, ao qual é introduzida uma classificação nos atributos segundo o fato deles poderem ser usados como chave de recuperação de registros ou não (fazendo um paralelo com o Modelo de Representação de Objetos, os

atributos que podem ser usados como chave de recuperação corresponderia a objetos, e os demais corresponderiam a comentários). Os autores descrevem então um sistema de multiprocessadores composto por um VAX 11/780 e dois PDP 11/44, ao qual foi dada a denominação de MBDS (Multi Backend Database System). Sobre esse sistema são então apresentadas duas características que os autores postulam dever ser respeitadas em qualquer implementação de máquinas de base de dados:

- quando a base de dados permanece a mesma, o tempo de resposta a consultas deve cair para aproximadamente a metade se o número de processadores genéricos e o número de dispositivos de armazenagem de dados da máquina for duplicado;
- quando o conjunto de dados da resposta a uma consulta usando o modelo relacional dobra, o tempo de resposta deve ficar aproximadamente constante se o número de processadores genéricos e o número de dispositivos de armazenagem de dados da máquina for duplicado.

Um problema que em arquiteturas especiais assume conotações diferentes daquelas usuais em sistemas convencionais, é o problema da consistência das informações quando uma transação já iniciada é interrompida, ou quando o sistema sofre uma queda de força. Agrawal trata desse problema em [AGR_85], percorrendo sobre as possibilidades existentes, e sugerindo um algoritmo baseado na existência de processadores especializados no acompanhamento da execução de transações pelos demais processadores. O autor apresenta também um estudo comparativo do efeito desse algoritmo na performance da máquina DBC.

V.4.2. - Projetos Recentes de Máquinas de Base de Dados.

Recentemente, várias máquinas de base de dados têm sido propostas, porém nenhuma delas tem ainda uma implementação completa, e nem (com exceção daquelas implementadas em arquiteturas convencionais, tal como a máquina FRENDA) existe previsão para disponibilidade comercial desses dispositivos. Um ponto a ser observado é que hoje já não se fazem projetos de máquinas de base de dados que não levem em conta a estrutura que os dados terão, descrição dos mecanismos de restrição da região do meio de armazenagem de massa que contém os dados de interesse, e os próprios algoritmos para a execução das operações necessárias. Um apanhado de projetos de máquinas de bases de dados surgidos recentemente é efetuado a seguir.

Uma proposta de desenvolvimento de máquina de base de dados que vem recebendo bastante atenção ultimamente é a Máquina de Base de Dados Relacional DELTA, em desenvolvimento para apoio ao projeto de Sistemas de Computação de Quinta Geração japonês. O objetivo desse projeto é construir uma máquina de base de conhecimento e uma máquina de inferência paralela, que deverão integrar-se para constituir o sistema de quinta geração pretendido, com capacidade de inferência e decisões baseadas em conhecimento armazenado. Na primeira fase do projeto, a máquina de base de dados relacional DELTA, junto com um conjunto de máquinas pessoais de inferência seqüencial PSI ([YOK_84], [TAK_84]), interligadas através de uma rede local de alta performance constituem o resultado das pesquisas ([FUC_84] e [KAW_84]) (Figura 27).

A máquina DELTA [TAK_85] consiste em um dispositivo que pode ser acessado através de uma rede local por um grande número de estações PSI, o qual mantém todos os dados disponíveis ao sistema em uma base de dados relacional. É constituída essencialmente de quatro tipos de unidades, tal como mostrado na Figura 28:

- um processador de interface, que cuida de toda a comunicação da máquina DELTA com a rede local;
- um processador de controle geral da máquina;
- de um a quatro dispositivos relacionais de base de dados (Relational Database Engine); e
- uma memória hierárquica.

Figura 27 - A Máquina DELTA em uma Rede Local.

Além disso, um quinto módulo denominado Processador de Manutenção supervisiona o estado de processamento de todos os demais, e está conectado a uma console, onde um operador pode se inteirar do estado de processamento da máquina e executar comandos de reconfiguração dos demais componentes.

O processador de controle recebe do processador de interface os pedidos de acesso à base de dados em uma linguagem baseada na

linguagem de consulta relacional SQL [AST_76], e a compila para uma seqüência de comandos que são enviados para os dispositivos relacionais. Estes são os processadores que efetuam as operações de manipulação de dados tais como criação, união, seleção e projeção de relações. As cadeias de dados processadas são lidas da memória hierárquica, e as relações resultantes são de novo reescritas na memória hierárquica, de onde podem ser repassadas às estações PSI através do processador de interface [SBY_84].

Figura 28 - Estrutura da Máquina de Base de Dados DELTA.

A memória hierárquica nada mais é do que uma combinação de um grande banco de memória cache, associada aos dispositivos de disco de alta capacidade, controlados por um processador dedicado que analisa todos os pedidos de dados de todos os demais módulos da máquina DELTA, e providencia a disponibilidade dos mesmos na memória cache. Dessa maneira, os demais processadores não necessitam conhecer a existência dos dispositivos de armazenagem de massa, mas apenas das unidades de memória cache.

Cada dispositivo relacional [SAK_84] é constituído de um processador convencional de 16 bits, dispendo de 512 KBytes a 1 MBytes de memória local, e associado a dois adaptadores para acesso de alta velocidade à memória hierárquica (um para leitura e outro para escrita de dados), e um arranjo de células de ordenação para facilitar as operações de união e projecção de relações.

Na Universidade de Illinois em Urbana, está em desenvolvimento uma máquina de base de dados [KGK_84] destinada a suportar o modelo de dados relacional através de uma arquitetura usando o conceito de "Pipeline". É proposto que sejam desenvolvidas pastilhas especiais para a construção dessa máquina, aproveitando o recente desenvolvimento nessa área, que permitiria grande integração das funções necessárias à essa máquina.

A proposta baseia-se no conceito que qualquer consulta à uma base de dados relacional, expressa em uma linguagem de alto nível, pode ser decomposta em um pequeno número de Operações Primitivas na Base de Dados, caracterizadas da seguinte maneira:

- Uma operação de consulta sobre apenas uma relação, e sem envolver construções tais como ORDENANDO POR, AGRUPANDO POR, SELECIONE UNICAMENTE, etc consiste de uma operação primitiva;
- Cada construção do tipo citado acima aumenta uma operação primitiva na quantidade de primitivas de uma consulta, a menos da construção ORDENADO POR, que pode corresponder a uma ou duas operações primitivas, dependendo do número de tuplas na relação resultante;

- operações de união e projeção são tratadas como se consistissem de apenas uma operação primitiva. Operações que envolvem a união de n relações são tratadas como se consistissem de $n-1$ operações primitivas.

A arquitetura proposta é constituída essencialmente de quatro processadores operando em "pipeline", cada um denominado um PIPE, e cada um especializado na execução de uma operação primitiva, e que são:

PIPE de Seleção de tuplas numa relação;

PIPE de Agregação;

PIPE de União;

PIPE de Ordenação;

A máquina é constituída então desses 4 PIPEs, uma rede de chaveamento, um processador (tradicional) que efetua a compilação de cada consulta nas operações primitivas que a compõem, um processador controlador das operações primitivas em execução, e um conjunto de memórias atuando como Filas para ajustar a entrada e saída dos vários processadores e dispositivos de entrada e saída. A rede de chaveamento permite que o resultado de qualquer PIPE possa ser submetido a qualquer PIPE para o prosseguimento das operações que constituem a consulta.

Desenvolvendo o conceito de "Superfície Associativa", Hahne e outros em [HAH_85] propõem uma arquitetura SIMD (uma só instrução operando sobre muitos dados) específica para SGBDs relacionais. A arquitetura consiste em um certo número de unidades lógico-aritméticas, denominadas PE (Processor Element), cada uma contando com três registradores de dados e lógica para

operações de comparação de um registrador com um dado recebido da memória, todas executando ao mesmo tempo uma mesma instrução. A memória é dividida em "fatias de Bytes" (Byte Slices), sendo que cada fatia é acessível a apenas um PE. Assim, em cada operação de comparação, são comparados simultaneamente um número de bytes igual ao número de PE. Ao conjunto de PEs os autores denominam Superfície Associativa.

Os autores descrevem uma máquina projetada com esses conceitos denominada "Hybrid Associative Store" - HAS, constituída de 64 PEs, e uma memória de 64 fatias de 256 KBytes cada uma, sendo construída nos Laboratórios de Pesquisa de Siemens AG em Munique.

Baseados no conceito de que um fator limitante no desempenho de uma máquina de base de dados utilizando múltiplos processadores é a sua capacidade de transferência de dados entre os processadores e os dispositivos de armazenagem de dados, Bic e Hartmann em [BIC_85] propõem uma arquitetura baseada no uso de muitas unidades de disco rígido de pequena capacidade, cada uma acoplada a um processador, os quais por sua vez estarão em uma malha de processadores, onde cada um pode se comunicar diretamente com quatro "vizinhos", formando um arranjo "quadrado" bidimensional de processadores. Cada processador é um processador genérico de 16 bits dispondo de memória local para dados de tamanho aproximadamente 3% do tamanho da capacidade do disco ao qual está associado.

Os autores propõem então um novo modelo de dados para tirar proveito dessa arquitetura, o qual denominam Modelo de Grafo Ativo (Active Graph model). Esse modelo nada mais é do que o próprio modelo entidade-relacionamento, onde tanto a estrutura

dos dados quanto os próprios dados são chamados de nós do grafo. Todos os nós correspondentes a dados estão conectados ao nó da estrutura ao qual aquele dado pertence. Sempre que alguma referência é feita a um determinado conjunto de entidades, esse nó é "ativado", o que causa a "ativação" de todos os nós de dados subordinados a esse nó; cada nó ativado, ativa por sua vez os nós de qualquer relacionamento ao qual esteja associado, e que tenha sido mencionado na consulta estabelecida, até que ao final estarão ativados os nós do conjunto de entidades que constitui a resposta à consulta solicitada.

Em termos teóricos, pode-se dizer que o modelo proposto constitui-se na própria Teoria das Redes de Petri [PET_77], porém aplicada aos dados, e não aos processos, tal como na teoria original. Comparando-se ao Modelo de Representação de Objetos, constitui-se na associação de atributos com características de Regras a objetos, de maneira que determinadas consultas possam prosseguir automaticamente, uma vez que uma determinada condição de acesso seja iniciada.

Aproveitando o barateamento das unidades de memória, Leland e Roome em [LEL_85] apresentam uma arquitetura que dispensa o uso de dispositivos de disco (usados apenas para "backup"), e sugerindo o uso de uma extensa memória de acesso aleatório (da ordem de 100 a 2000 Mbytes de dados, denominada Memória Estável (Stable Memory - SM). Para atender ao processamento do SGBD, os autores sugerem da ordem de 3 a 16 processadores.

Existem três tipos de processadores, todos de idêntica constituição, porém com atribuições de processamento diferentes:

um Gerente de Consulta, que recebe cada consulta, faz a sua compilação, e aloca os demais processadores para executar cada operação de que se constitui a consulta;

Processadores de Consulta, que são responsáveis pela manutenção da consistência da base de dados como um todo, para cada operação em execução sob a sua supervisão, bem como da consistência dos dados envolvidos nessas operações; e

Gerentes de Relações, que executam operações sobre uma relação ou parte de uma relação em colaboração com outros Gerentes de Relações.

Tendo todos a mesma constituição, cada processador pode assumir qualquer das atribuições, permitindo uma configuração do sistema ajustada a cada caso em particular. Porém, uma reconfiguração somente poderia ser efetuada com a base de dados em um estado consistente, sem nenhuma operação de consulta em andamento, e sob explícita intervenção de um operador.

Numa implementação desse tipo, o ponto crítico constitui-se no acesso a essa memória; uma vez que se essa fosse toda a memória do sistema, a concorrência de acesso a esses dados seria elevadíssima. Assim, os autores propõem que cada processador disponha de memória própria, que por sua vez estaria dividida em duas partes: Memória Local, onde os programas e dados genéricos estariam armazenados; e Memória de Comunicação, onde as relações de dados sendo manipuladas localmente estariam disponíveis, sem que fosse necessário acesso à Memória Estável. Dessa forma, a Memória de Comunicação atuaria tanto como memória cache, para as operações sendo realizadas localmente, quanto como uma memória de segurança, evitando que alterações fossem feitas na base de dados

antes que uma transação houvesse sido satisfatoriamente completada.

Uma solução intermediária para evitar o uso apenas de memória de acesso aleatório, e ainda assim dispôr de toda a facilidade e flexibilidade de acesso provida por essa organização, Browne e outros [BRD_85] propõem uma arquitetura em que cada dispositivo de armazenagem de dados estaria associado a um processador e memória local, onde o acesso aos dispositivos seria feito emulando um espaço de endereçamento contínuo, tal como se fora uma memória de acesso aleatório. Cada processador responderia ao espaço de endereçamento para o qual a unidade de disco a si associada corresponde, e atuaria no sentido de manter em sua memória local cópias dos registros sendo solicitados com maior freqüência, fazendo assim sua memória atuar como memória cache para o dispositivo de armazenagem, e como memória virtual paginada para os computadores executando os programas de aplicação.

Nessa proposta, os autores consideram que possa haver vários computadores acessando essa máquina simultaneamente, através de uma malha de interconexão. A estrutura dessa malha corresponderia a de árvores, que poderiam permitir o acesso de qualquer computador a qualquer dispositivo de armazenagem, sem redundância de caminhos de acesso, e garantindo ausência de conflito nos caminhos de acesso.

Uma das operações mais caras do modelo relacional, em termos de tempo de execução e consumo de recursos computacionais, é a operação de união de duas relações. Existem vários algoritmos básicos para a sua execução, e conforme foi mostrado por

Valduriez e outros em [VAL_84], cada algoritmo é melhor para cada caso quando se trata da execução em um ambiente de múltiplos processadores, dependendo do número de processadores envolvidos, do tamanho de cada relação original, e da proporção de tuplas que têm correspondência nas duas relações. Os autores implementaram três algoritmos na máquina SABRE, a saber:

Algoritmo de dois "loops" de comparação aninhados;

Algoritmo de ordenação parcial e mistura (Sort-Merge);

Algoritmo de partição para união (Hashing-join).

A seguir, fazem a avaliação comparativa dos mesmos, e estabelecem-se os critérios que podem orientar qual o melhor algoritmo em cada caso particular. Os autores propõem que uma máquina deve ser projetada levando em conta que todos os algoritmos deveriam ser igualmente bem executados, uma vez que mesmo em arquiteturas dedicadas o custo da operação de união obriga a busca da melhor opção para a execução dessa operação.

Em [HON_85] Hong propõe uma arquitetura que permite a execução de uma variante do Algoritmo de Partição para União, dotando-a de uma estrutura de memória capaz de marcar cada tupla através de um arranjo linear de bits. Nessa arquitetura, todas as tuplas de uma relação são pesquisadas seqüencialmente para se determinar a existência de tuplas que se ajustam aos critérios de busca especificado pela operação de união solicitada. Todas as tuplas que se ajustam tem o seu correspondente bit no arranjo linear marcado. A seguir a mesma operação é efetuada sobre a segunda relação. Da intersecção dos dois vetores de bits resulta a união. Para que o processo possa ser efetuado, as características da operação solicitada são analisadas, para se escolher um

algoritmo adequado. Assim, a operação de união é classificada em:

União implícita - onde a relação resultante obtém os dados de apenas uma das relações originais;

União explícita tipo I - onde existe o relacionamento um-para-muitos entre as duas relações originais;

União explícita tipo II - onde existe o relacionamento muitos-para-muitos entre as duas relações originais.

A determinação de a qual bit corresponde cada tupla pesquisada é feita dividindo-se a relação como um todo em um certo número de blocos, sendo que cada bloco é submetido a um processador de um conjunto deles (denominados Servidores de Tuplas). O critério de separação em blocos é baseado no domínio do atributo usado para a união das duas relações, caracterizando assim a operação de partição da relação.

Considerando que uma máquina de base de dados não necessite necessariamente ser constituída de uma arquitetura admitindo processamento paralelo, mas tão somente que deve ser um dispositivo que realize todas as operações associadas a máquinas de base de dados, Hikita e outros em [HIK_85], propõem a máquina FREND (Flexible Relational Database Machine), baseada em um microprocessador de 16 bits que realiza todas as operações de gerenciamento de dados solicitadas por estações de trabalho a ela conectadas através de interfaces seriais de dados. Essa abordagem não é tradicionalmente caracterizada como uma máquina de base de dados, mas sim como um sistema distribuído onde as operações de gerenciamento de dados estão centralizadas. Porém os autores justificam essa colocação argumentando que essa máquina, além de centralizar todas as operações de gerenciamento de dados, não

realiza além dessa nenhuma outra tarefa.

V.5. - Conclusões.

Pode-se notar que a partir do surgimento do conceito de Máquinas de Base de Dados no início dos anos 70, ocorreram três fases de desenvolvimento: o desenvolvimento de Máquinas baseadas em discos associativos; máquinas que se utilizaram de lógica adicional para a execução das operações da máquina, empregando notadamente as arquiteturas SIMD; e atualmente, a fase em que as máquinas procuram aumentar a eficiência dos SGBDs tanto através da existência de lógica adicional quanto através do desenvolvimento conjunto de lógica física e algoritmos especiais para a arquitetura.

Desde o seu início, quando bases de dados empregando qualquer modelo de dados eram igualmente tratadas, até os trabalhos atuais, o modelo de dados relacional tem se mostrado o predileto para os projetistas das máquinas de base de dados, chegando mesmo a ocorrer que hoje se considera padrão o fato de que todas as máquinas de base de dados tratam com exclusividade o modelo relacional. Mesmo em casos de máquinas que têm por objetivo o modelo Entidade-Relacionamento [BIC_85], as operações básicas supridas correspondem fundamentalmente àquelas de seleção, união e projeção de relações.

O desenvolvimento de uma máquina de base de dados deve levar em conta três aspectos principais:

- a arquitetura da nova máquina;
- o modelo de dados que a máquina deve suportar;
- o conjunto de algoritmos que irão ser executados, asso-

- ciados ao modelo de dados que a máquina deve suportar;
- o esquema de proteção que o sistema apresenta quanto a interrupções forçadas de execução e à manutenção da integridade física e lógica dos dados.

Os esquemas de proteção adotados pelas várias máquinas são muito pouco estudados, sendo que em apenas uma delas (RAP.2 - [SCH_79]) existe a preocupação de se cuidar desse aspecto. O reconhecimento dessa falha, feito em [AGR_85], não tem por enquanto obtido resposta por parte dos projetistas de máquinas.

Nos dois capítulos seguintes deste trabalho apresentam-se respectivamente uma máquina já construída e considerações sobre uma máquina que deve suportar o Modelo de Representação de Objetos (MRO). O estudo sobre as máquinas de base de dados existentes, efetuado neste capítulo tem como objetivo o reconhecimento das técnicas usualmente empregadas para desenvolvê-las. No entanto, as características desse modelo impõem para uma máquina que o suporte necessidades bastante diversas daquelas impostas por um SGBD relacional. Dessa forma, os algoritmos usualmente estudados nessa área são de pouca ou nenhuma valia para o seu emprego em uma máquina que suporte o MRO, visto que as operações executadas por um SGBD relacional não são usualmente utilizadas quando se emprega o MRO. Devido a isso, nesse capítulo procurou-se dar destaque às arquiteturas implementadas nas várias máquinas existentes, ao passo que os algoritmos usados foram apenas superficialmente apresentados.

A ARQUITETURA DA MÁQUINA IMPLEMENTADA

VI.1. - Introdução.

O Sistema de Apoio por Computador à Documentação de Sistemas - SACDS - descrito em [TRA_82], utiliza-se de um Sistema de Gerenciamento de Bases de Dados concebido e desenvolvido especialmente para ele. Esse SGBD apoia-se em um modelo de dados também especialmente desenvolvido, que pode ser considerado como o precursor do Modelo de Representação de Objetos, pois a partir dele os conceitos apresentados neste trabalho foram desenvolvidos. Aquele modelo não foi dada nenhuma designação, e por isso ele será identificado no restante deste trabalho apenas como o "Modelo de Dados do SACDS". Naquele trabalho, sugeria-se que o SACDS (Sistema de Apoio por Computador à Documentação de Sistemas), beneficiar-se-ia da disponibilidade de um sistema de processamento de dados que pudesse executar algumas tarefas a nível do SGBD, independentemente (em paralelo) das tarefas do próprio SACDS. Assim, surgiu o interesse no desenvolvimento de uma máquina que permitisse aproveitar esse paralelismo.

A motivação para o desenvolvimento dessa máquina foi a de separar as tarefas de gerenciamento da base de dados do processador que estivesse executando uma ferramenta do SACDS. Dessa

forma, a solução seria o desenvolvimento de uma máquina que dispusesse de um processador dedicado às tarefas do SGBD, e outro para executar o próprio SACDS. Como o acesso a disco é vital para a execução do SGBD, e ao mesmo tempo o SACDS, se for desconsiderada a sua necessidade de acesso a Base de Dados, necessita muito pouco acesso a disco, o processador que desempenhasse a função de gerenciamento da base de dados deveria ser também o controlador das unidades de disco e disquete do sistema.

Para que o grande volume de Software necessário tanto ao SACDS quanto ao seu SGBD pudesse ser adaptado à nova máquina com relativa facilidade, seria interessante que os dois processadores apresentassem um ambiente de programação ameno para um sistema operacional já existente, do qual pudessem ser aproveitados editores e compiladores já existentes. Assim, cada um dos processadores foi projetado para permitir executar o Sistema Operacional CP/M [DIG_79a] [DIG_79b].

Foi considerado que seria interessante (e que não apresentaria dificuldades adicionais) a disponibilidade de mais um processador no sistema, que pudesse servir para acompanhar-se o andamento da execução dos outros dois, e optou-se para que esse terceiro processador fosse semelhante ao processador que executaria o próprio SACDS, uma vez que as necessidades de ambos seriam semelhantes (sob o ponto de vista do Sistema Operacional, pois o SACDS consiste de programas de aplicação comuns, e o terceiro processador executaria também programas igualmente comuns). Assim, efetivamente somente dois tipos de processadores existiriam, e considerou-se que o sistema poderia ser desenvolvido para contar não com dois processadores do tipo que executaria o próprio SACDS, mas com um número maior. Considerou-se que quatro

desses processadores seria um bom número, pois por um lado aproveitaria melhor o processador capaz de controlar as unidades de disco e disquete. e por outro, um número maior de processadores não poderia mais ser atendido com rapidez por esse mesmo processador.

O sistema que foi então implementado, denominado Máquina SACDS/SGBD, é apresentado a seguir. O modelo de dados do SACDS é apresentado na seção VI.2. Na seção VI.3 é feita a apresentação geral dessa máquina, e como ela pode ser usada para suprir as necessidades de processamento do SACDS, permitindo a execução em paralelo do SGBD e do Sistema de Apoio ao Projeto de Sistemas. Na seção VI.4 é descrita a forma como os processadores que compõem essa máquina se comunicam, mostrando-se como é possível implementar as primitivas de acesso à Base de Dados através das primitivas de comunicação entre os processadores, e como o processador que executa o programa de aplicação pode usar apenas primitivas de alto nível para o acesso à Base de Dados. Na seção VI.5 é descrito o uso que se faz da memória dessa máquina para o processamento do SGBD para o SACDS, e na seção VI.6 é feita a descrição de como os arquivos de dados necessários para implementar essa Base de Dados são tratados, de maneira a permitir um uso eficiente da arquitetura implementada.

VI.2. - O Modelo de Dados do SACDS.

O modelo de dados do SACDS pode ser visto como um modelo desenvolvido a partir do Modelo Entidade-Relacionamento, ao qual foi feita a extensão de caracterizar os atributos como sinônimos

ou como comentários, e realizar a Estrutura de Dados Independente do Caminho de Acesso da mesma maneira como é feita no MRO. Na implementação feita para o SACDS não foi prevista a associação de atributos de nenhuma característica a relacionamentos, sendo que apenas atributos com características de comentário e atributos com características de sinônimo foram previstos para os objetos. Na Figura 29 está a representação de um Meta-Eschema de Dados que corresponde ao Modelo de dados do SACDS.

A cada uma das entidades representadas nessa figura corresponde um arquivo de dados na Base de Dados. Além disso, existe um outro arquivo que mantém os parâmetros de estado dos outros quatro arquivos, e o SGBD acessa um sexto arquivo que contém as informações sobre os tipos de objetos, de relacionamentos e de comentários, denominado Arquivo de Definição de Linguagem, pois especifica as palavras chaves com que uma descrição é feita, bem como a sintaxe dessas palavras. O Arquivo de Definição de Linguagem do SACDS armazena informações que correspondem à Meta-base de dados no Modelo de Representação de Objetos. O arquivo de parâmetros armazena o estado de uma base de dados num dado instante, e os acessos a ele são efetuados apenas no início e no fim de cada sessão de atividades do SACDS, motivo pelo qual o SGBD em operação normal necessita apenas dos outros cinco arquivos.

O SACDS pode ser caracterizado como um sistema automatizado de Projeto Auxiliado por Computador (PAC), que embora tenha sido desenvolvido tendo como objetivo principal a Engenharia de Software, sempre manteve o enfoque de ser uma ferramenta que pudesse ser utilizado para a descrição de sistemas em geral, e não apenas dos sistemas de software. Dessa forma, caracteriza-se

Figura 29 - Meta-Esquema de Dados da Base de Dados do SACDS.

como um Sistema de Gerenciamento de Dados para Engenharia, e compartilha com aqueles as suas necessidades.

VI.3. - A Configuração da Máquina SACDS/SGBD.

Já nas conclusões de [TRA_82], e em atividades desenvolvidas posteriormente, pôde-se perceber que as atividades de manipulação de dados envolvidas na execução do SACDS davam margem ao processamento simultâneo e independente do próprio SACDS e do SGBD por ele utilizado, e além disso, o próprio SGBD ofereceria vantagens se pudesse ser executado em máquinas que pudessem executar processos utilizando multiprocessamento.

A partir daquele trabalho, foi montada uma máquina que permitisse avaliar os benefícios da execução do SACDS e do SGBD por ele utilizado em processadores separados. Inicialmente foi prevista para essa máquina a execução de uma tarefa do SACDS de cada vez, em uma estação de trabalho, e a possibilidade de que outras estações executassem outras tarefas não pertencentes ao SACDS, num total máximo de até quatro estações de trabalho. A unidade de processamento que cuidaria das tarefas de Gerenciamento da Base de Dados seria responsável também por todo o gerenciamento de arquivos em disco e controle da impressora do sistema. A Figura 30 mostra esquematicamente a configuração dessa máquina.

Essa máquina foi construída tendo como objetivo permitir que qualquer estação de trabalho pudesse executar as tarefas pertencentes ao SACDS, mas somente uma estação de trabalho podendo fazê-lo ao mesmo tempo. As estações de trabalho que num dado instante não estivessem executando as tarefas do SACDS poderiam executar outras tarefas usuais de Processamento de Dados, tal como edição de arquivos, compilação, etc, para as quais o Processador de Gerenciamento Central atua apenas como um controlador das unidades de Disco, disquetes e impressora.

Figura 30 - Configuração da Máquina SACDS/SGBD.

A comunicação entre os processadores foi efetuada de modo a permitir que qualquer estação de trabalho pudesse se comunicar com o Processador de Gerenciamento Central, através de um conjunto de instruções de alto nível. Não existe nenhuma previsão para uma comunicação direta entre duas estações de trabalho.

Para que essa máquina pudesse dispor de uma ampla gama de software, foi estruturada de maneira que cada estação de trabalho apresentasse um ambiente compatível com o de outras máquinas já existentes. Como se trata de uma máquina experimental, a escolha de qual processador, e qual ambiente de programação usar, foi

orientada mais por considerações sobre a disponibilidade dos recursos para implementá-la e avaliá-la do que sobre considerações sobre os melhores processadores então existentes no mercado. Devido a isso, foi escolhido o processador Z80, e o Sistema Operacional CP/M-80.

Cada Estação de Trabalho é constituída por um micro-processador Z80, com 64 KBytes de memória disponível, e um Porto Serial de Comunicação para o terminal de vídeo da Estação de Trabalho. Cada Estação de Trabalho pode executar programas sob o Sistema Operacional CP/M, ou o SACDS. Apesar de na versão atual não haverem sido implementados outros periféricos, é possível que cada Estação de Trabalho tenha acesso a outros periféricos particulares, tais como unidades locais de disquetes ou outros portos de comunicação.

O Processador Central de Gerenciamento é constituído de um micro-processador Z80 com até 512 KBytes de Memória RAM, e 28 KBytes de Memória ROM. Esse processador normalmente executa o Software Básico de Comunicação com as Estações de Trabalho e controle dos Periféricos Centralizados, porém pode opcionalmente executar também o Sistema Operacional CP/M, com o objetivo de executar programas de supervisão do sistema. Quando isso não é necessário, não é necessária a existência de um terminal de vídeo conectado à seu Porto Serial.

Dessa forma, cada Estação de Trabalho pode executar a ampla gama de Software existente para o Sistema Operacional CP/M, e as tarefas de desenvolvimento de Software para essa máquina puderam ser restritas ao desenvolvimento do Software Básico para tornar a máquina funcional, tal como o monitor e os controladores de periféricos, ao software para a comunicação entre os processado-

res, e ao Software de Gerenciamento da Base de Dados adaptado para a execução em separado do SACDS e do SGBD, uma vez que os programas de edição, compilação e utilitários podem ser os já existentes para máquinas que executam o CP/M.

Através dessa concepção, o hardware e o software básico que constituem essa máquina podem ser aproveitados como um computador multiusuário de propósito geral baseado em micro-computadores, que apresenta como vantagem sobre um conjunto de vários micro-computadores o fato de manter compartilháveis a impressora e as unidades de disco e disquetes - os periféricos mais caros de uma configuração de micro-computador. Assim, a máquina desenvolvida, antes de ter disponível o próprio SACDS, tornou disponível como subproduto de seu desenvolvimento uma máquina que é útil para outras áreas da informática.

VI.4. - A Comunicação entre os Processadores da Máquina SACDS/SGBD.

As Estações de Trabalho da máquina implementada comunicam-se com o Processador Central através de um Porto Paralelo de 8 bits de dados, cujo hardware é sumariamente descrito a seguir. O protocolo lógico de comunicação implementa instruções de alto nível, sendo que cada instrução sempre origina-se em uma Estação de Trabalho, e é então atendida pelo Processador Central.

A comunicação é efetuada através de uma pastilha 8255 em cada Estação de Trabalho, operando no modo de programação 2/input, ou seja, o Porto A é usado como Porto Bidirecional, o Porto B é usado como Porto de entrada, e todas as linhas do Porto

C são usadas como linhas de controle. A pastilha 8255 está associada ao hardware da Estação de Trabalho, porém é controlada pelo Processador Central. A Figura 31 mostra como a pastilha 8255 é usada para a comunicação entre cada Estação de Trabalho e o Processador Central.

Figura 31 - Comunicação entre o PCG e uma Estação de Trabalho.

O Processador Central atende as Estações de Trabalho em regime de Varredura ("Polling"), verificando ciclicamente o estado do Porto C de cada uma das quatro pastilhas 8255 de comunicação a que tem acesso. Uma estação de trabalho inicia uma instrução de comunicação colocando o código da instrução desejada no porto B da pastilha 8255 que lhe está associada, e aguardando que ela seja lida pelo processador central. A seguir, prossegue através de um protocolo síncrono de comunicação, trocando dados com o processador central através do Porto A. A sequência de

entrada ou saída de dados (do ponto de vista de cada um dos processadores) é determinada pelo código da instrução em andamento.

Para manter uma taxa de transferência de dados elevada, é usada a técnica de interrupção baseada na linha de interrupção não mascarável ("-NMI") do processador Z80 e na geração de sinais de pedido de dados ("Data Request"), e pedido de finalização de transferência ("Interrupt Request") desenvolvida em [MAR_83]. Dessa forma, pode-se atingir uma taxa de transferência de até 130 KBytes por segundo. O sinal de "Data Request" é usado para manter o sincronismo da transferência entre os dois processadores, ao passo que o sinal de "Interrupt Request" é usado para garantir que um processador não possa "travar" o outro devido a alguma falha ou perda de sincronismo que possa ocorrer.

Os sinais de "Data Request" e "Interrupt Request" enviados de cada Estação de Trabalho para o Processador Central são filtrados por uma série de portas controladas pelo Processador Central, para evitar interferência de sinais simultâneos de mais de uma Estação de Trabalho. Além disso, todos os sinais de "Data Request" e de "Interrupt Request" são mantidos indefinidamente através de biestáveis, liberando cada processador em comunicação de verificar sua aceitação. O circuito completo empregado para a comunicação entre uma Estação de Trabalho e o Processador Central é mostrado na Figura 32.

O conjunto de instruções de comunicação entre uma Estação de Trabalho e o Processador Central pode ser dividido em dois subconjuntos: as instruções de acesso aos periféricos; e as instruções de acesso à Base de Dados. As instruções do segundo

a) Circuito de comunicação do Processador Central de Gerenciamento.

b) Circuito de Comunicação de uma Estacao de Trabalho.

Figura 32 - Circuito Completo da Comunicação entre Processadores

subconjunto somente podem ser enviadas por uma Estação de Trabalho de cada vez.

Todas as instruções seguem uma sequência de três fases padrão. A primeira fase corresponde à inicialização da comunicação, quando a Estação de Trabalho coloca no Porto B da sua pastilha 8255 o código da instrução solicitada. Uma vez lido esse código pelo Processador Central, a segunda fase é efetuada, a qual corresponde à transferência dos dados da instrução solicitada pela Estação de Trabalho para o Processador Central. O final da segunda fase ocorre quando todos os bytes tiverem sido transferidos, e ambos os processadores enviam para o outro sinais de "Interrupt Request". A seguir, a Estação de Trabalho deve aguardar a resposta do Processador Central, o qual estará providenciando a execução da instrução recebida. Uma vez obtidos os dados solicitados, a terceira fase é executada, com o Processador Central enviando esses dados para a Estação de Trabalho, que havia ficado esperando por eles. O primeiro byte transferido nessa fase é sempre um byte de código indicando o estado final da transferência, sendo o valor "zero" sempre usado para indicar que a instrução solicitada foi realizada sem que ocorresse nenhum erro ou situação especial.

O número de Bytes transferidos nas fases dois e três é determinado pelo código da instrução em andamento, e é sempre igual para uma mesma instrução.

A implementação realizada obriga que a Estação de Trabalho fique parada enquanto o Processador Central obtém os dados a serem transmitidos na terceira fase da instrução. Isso decorre do fato de não haver sido empregado um circuito de Acesso Direto à Memória ("DMA"), que poderia cuidar da transferência enquanto o

processador da Estação de Trabalho continuaria a executar a sua tarefa. Esse circuito não foi empregado porque seria mais dispendioso do que o circuito adotado, ao mesmo tempo em que não traria benefício algum ao sistema, pelo simples fato de que a Estação de Trabalho executa somente um programa de cada vez (não existe multiprogramação em um computador executando o Sistema Operacional CP/M e nem executando uma ferramenta do SACDS). Dessa forma, mesmo que pudesse ser liberada para continuar o processamento, a Estação de Trabalho teria de continuar aguardando o resultado da operação solicitada, pois não teria mais nada para fazer.

VI.4.1. - As Instruções de Acesso aos Periféricos Centralizados.

A estrutura do Sistema Operacional CP/M centraliza todas as operações sobre os periféricos em um módulo de Software denominado "Basic Input-Output System - BIOS". Assim, todas as operações de acesso a qualquer periférico do sistema são centralizadas numa Rotina de Manipulação de Periférico, que executa todas as funções possíveis sobre aquele periférico. As operações de acesso aos periféricos centralizados são executadas pelas rotinas do BIOS que cuidam do acesso a Discos e as que cuidam de acesso à impressora.

As instruções de acesso aos Periféricos Centralizados são as que possuem códigos de 1 a 5, porém estão reservados os números seguintes até 64 para expansão futura. A seguir são mostrados os significados dos bytes transferidos em cada uma das Instruções de Acesso aos Periféricos Centralizados implementadas:

Fase 1 - Código 1 = Leitura de um registro de 128 Bytes do Disco.

Fase 2 -

Fase 3 - Estado da leitura e os 128 Bytes do registro lido.

Fase 1 - Código 2 = Escrita de um registro de 128 Bytes no Disco.

Fase 2 -

Bytes 6 a 133 Registro a ser Escrito.

Fase 3 - Estado da Escrita.

Fase 1 - Código 3 = Escrita de um Caracter na impressora.

Fase 2 - Byte a ser impresso.

Fase 3 - Estado da Escrita.

Fase 1 - Código 4 = Informa disponibilidade de impressora.

Fase 2 -

Fase 3 - Informa disponibilidade ou não.

Fase 1 - Código 5 = Solicitação de Acesso à Base de Dados.

Fase 2 -

Fase 3 - Informa autorização ou rejeição.

As quatro primeiras instruções acima são suficientes para a execução do Sistema Operacional CP/M, para cuja execução foram especialmente realizadas. O sistema operacional CP/M gerencia o uso de unidades de disco e disquete em um sistema com apenas um processo em execução simultânea, e não prevê a possibilidade que mais de um processo possa utilizar de maneira concorrente a ele a mesma unidade de disco. Tal situação impede que mais de uma Estação de Trabalho possa ter acesso para escrita a uma mesma unidade de disco. A solução para evitar isso seria através do desenvolvimento de um novo Sistema Operacional que apresentasse a

nível de interface com os programas de aplicação o mesmo padrão que o apresentado pelo CP/M, porém que dispusesse de capacidade para gerenciar a concorrência imposta pela execução simultânea de multiprocessadores. Prevendo o desenvolvimento de um tal sistema operacional, os códigos de instrução de 6 até 64 estão reservados para isso.

A instrução de número 5 é empregada para verificar a disponibilidade do Processador Central para operações como um Sistema de Gerenciamento de Base de Dados para que a Estação de Trabalho possa executar o SACDS. Essa instrução informa se já existe outra Estação de Trabalho executando o SACDS, e se não há, então coloca a Estação de Trabalho apta a executá-lo, bloqueando qualquer solicitação de outras estações.

VI.4.2. - As Instruções de Acesso à Base de Dados.

O acesso à Base de Dados é efetuado através de um conjunto de instruções, que permitem à Estação de trabalho solicitar do SGBD todas as operações necessárias. Apenas um pequeno conjunto de instruções foram implementadas, porém a forma de comunicação de todas as instruções necessárias seguem o mesmo padrão, que é apresentado a seguir através de um exemplo.

Considere-se a operação "Criar um objeto de determinado Tipo e colocar o seu nome na Base de Dados". Essa operação pode ser solicitada através da chamada de uma função, que teria como argumentos de entrada o Nome e o Tipo desse objeto, e como argumento de saída a informação do fato desse objeto poder ser

colocado, ou condições de erro que impedissem a sua colocação:

ColocaObjeto (Nome, Tipo, Erro)

Como consequência de uma operação bem sucedida, a operação imporá o novo objeto como Objeto Corrente, além de colocá-lo na base de dados. Operações subseqüentes na base de dados admitiriam ser esse objeto o envolvido naquelas operações, até que uma nova operação indicasse um outro objeto como Objeto Corrente. Essa operação é solicitada para o Processador Central, agora executando o SGBD, através de uma instrução, cujo código indica a operação de Colocar-Objeto, e são enviados como dados da segunda fase da instrução o Nome e o Código do Tipo do Objeto. O Processador Central analisaria esses dados, obtendo o mais rapidamente possível a informação sobre a possibilidade de sua colocação, e executa a terceira fase da instrução, enviando o código de Erro (ou de operação sem erro) como resposta.

A instrução estará então encerrada, e o programa em execução na Estação de Trabalho poderá prosseguir. O Processador Central poderá então providenciar a efetiva colocação do objeto na Base de Dados.

Deve-se notar que dessa forma, tanto o Processador Central quanto a Estação de Trabalho poderiam executar cooperativa e paralelamente a tarefa de execução do SACDS. Além disso, o Processador Central poderia identificar as operações que decorrem da operação solicitada, tal como indicar para verificação os dados alterados, geração de seleções de objetos, etc, e executá-las quando não houvesse nenhuma solicitação das Estações de Trabalho.

Efetivamente, nem todas as operações podem ser executadas em paralelo. Por exemplo, instruções para obtenção de dados tal como "Obter o Nome de um objeto, dado o seu código" terão a sua segunda fase constituída da transmissão do código do objeto, e a terceira fase constituída do envio do código de erro e do nome do objeto. Para isso, é necessário que antes de iniciar a terceira fase, o Processador Central realize todo o processamento solicitado pela instrução, qual seja, o de localizar o nome da entidade indicada, não restando nenhum processamento para ser realizado após o encerramento da instrução. Porém, em muitos casos o processamento solicitado pode ser acelerado, se puder ser previsto quais informações a Estação de Trabalho irá solicitar a seguir, e buscando-se essas informações antecipadamente. Por exemplo, pode-se buscar o máximo de informações disponíveis na base de dados sobre o objeto corrente, sempre que o Processador Central não tenha que atender a solicitações de alguma Estação de trabalho.

VI.5. - A Memória dos Processadores da Máquina SACDS/SGBD.

Os processadores usados nas Estações de Trabalho e no Processador Central de Gerenciamento são micro-processadores Z80 de 8 bits, que têm o espaço de endereçamento de 64 KBytes. Nesse espaço de endereçamento devem ser executados os programas destinados a cada processador, e devem ser armazenados todos os dados e informações necessários a cada programa.

Em uma Estação de trabalho deve ser possível executar pro-

gramas operando sob o Sistema Operacional CP/M (que idealmente deve poder contar com todos os 64 kBytes), ou uma ferramenta que integra o SACDS. Além disso, todas as rotinas locais para a comunicação com o Processador Central devem estar permanentemente disponíveis.

No Processador Central, deve ser possível executar as rotinas de manipulação de periféricos e as rotinas centrais para a comunicação com as Estações de Trabalho, além de se poder manter nessa memória uma grande quantidade dos dados dos processos em execução nas Estações de Trabalho.

VI.5.1 - A Memória de uma Estação de Trabalho.

Para permitir que mais do que 64 KBytes de memória central pudessem ser acessadas por um dado processador, um esquema de chaveamento de bancos de memória foi implementado. A Figura 33 mostra esquematicamente como a memória de uma Estação de Trabalho está organizada.

Existe um Banco de Memória de Escrita e Leitura composto por duas páginas de 32 KBytes cada uma, denominadas RAM-0 e RAM-1, mais uma página de 32 KBytes denominada ROM, composta por um bloco de 4 KBytes de Memória de Acesso de Leitura e Escrita (RAM-e), um bloco de 4 KBytes de Memória de Apenas Leitura (ROM-p), e um bloco de 24 KBytes de Memória de Apenas Leitura (ROM-0).

A página ROM somente pode ser acessada na faixa de endereço de 0 a 32K, de maneira que quando essa página está ativada, ela

Figura 33 - Esquema de Memória de uma Estação de Trabalho.

ocupa a primeira metade do espaço de endereçamento do processador, sendo a segunda metade ocupada pela página RAM-0 ou RAM-1. Quando a página ROM não está ativada, as páginas RAM-0 e RAM-1 ocupam cada uma delas metade do espaço de endereçamento do processador, podendo a qualquer momento, qualquer uma delas ocupar a primeira ou a segunda metade desse espaço.

A página ROM tem o espaço de 0 a 4K ocupado pelo bloco ROM-p, o espaço de 4 a 28K ocupado pelo bloco ROM-0, e o espaço de 28 a 32K ocupado pelo bloco RAM-e. O bloco ROM-p contém todas as rotinas de comunicação com o Processador Central, incluído-se aí todas as rotinas para acesso a disco, impressora e operação com o SGBD, e todas as rotinas de manipulação do Porto de Comunicação Serial para o terminal de vídeo da Estação de Trabalho. O bloco ROM-0 não está sendo utilizado atualmente para nada, porém, poder-se-ia utilizá-lo para expandir as funções atualmente conti-

das no bloco ROM-p. O bloco RAM-e é constituído de memória de escrita e leitura que é utilizada para armazenar os dados necessários à execução das funções do bloco ROM-p. Atualmente apenas 2 KBytes dessa memória estão efetivamente usados.

Figura 34 - Circuito de Controle da Memória RAM de uma ET.

A Figura 34 apresenta o circuito de controle da memória de uma Estação de Trabalho. Para a memória do banco de escrita e leitura foram empregados oito circuitos integrados tipo 4164. Sendo uma memória dinâmica, é efetuado o revigoramento ("Refresh") usando a facilidade para isso oferecida pelo próprio processador Z80. Para a memória de apenas leitura foram empregados circuitos integrados tipo 2732, e para o bloco RAM-e foi empregado o circuito integrado tipo 6116, que sendo uma memória estática não necessita de revigoramento.

O uso desse esquema de memória por uma Estação de Trabalho é

o seguinte: quando a operação do sistema é inicializada, é ativada a página ROM, e é executada uma rotina de inicialização nela gravada. Essa rotina de inicialização coloca na segunda metade do espaço de endereçamento a página RAM-1, e nela carrega o BIOS do Sistema Operacional CP/M, solicitando-o do Processador Central. Tal como é preconizado pelo próprio CP/M, esse carregamento é feito de maneira que o sistema ocupe os endereços mais altos do sistema. Após a carga, a rotina de inicialização na página ROM passa o controle para a rotina de inicialização (CBOOT) do BIOS. O BIOS desse sistema está escrito de forma a que todas as suas funções são executadas através da ativação da página ROM, e a chamada de uma rotina nessa página que efetua a operação desejada. Dessa forma, o BIOS de uma Estação de Trabalho é constituído de um conjunto de rotinas que ativam a página ROM e chamam a rotina correspondente existente no bloco ROM-p dessa página. A rotina de inicialização do BIOS (CBOOT) carrega o restante do Sistema CP/M (BDOS e CCP), solicitando-o através da rotina de leitura do próprio BIOS (que o faz solicitando a leitura através da rotina correspondente no bloco ROM-p). Uma vez completado o carregamento, essa rotina desativa a página ROM, colocando na primeira metade do espaço de endereçamento a página RAM-0. O controle é então passado ao CCP e dessa forma o Sistema Operacional CP/M está ativo.

Sempre que é solicitada uma operação de leitura ou escrita em disco, a rotina correspondente do BIOS reativa a página ROM, e passa o controle para a rotina correspondente nessa página. Essa rotina faz a solicitação do setor lógico correspondente para o Processador Central. O endereço na memória onde o setor indicado está (ou será colocado) determina qual das páginas RAM-0 ou RAM-1

deverá ser ativada na segunda metade de endereçamento para que a transmissão com o Processador Central possa ser efetuada. Completada a comunicação, o controle volta à rotina do BIOS, que desativa a página ROM.

Com esse esquema de chaveamento de páginas de memória, cada Estação de Trabalho apresenta para o Sistema Operacional CP/M um ambiente de execução de programas que é normalmente identificado como sistema CP/M de 64 KBytes de memória, oferecendo para os programas de aplicação a maior quantidade de memória disponível em um ambiente CP/M.

As ferramentas que compõem o SACDS serão executadas nesse sistema como um programa CP/M normal, com a diferença que poderão usar um conjunto estendido de rotinas do BIOS, que suprem as funções de comunicação com a Base de Dados gerenciada pelo Processador Central.

VI.5.2. - A Memória do Processador Central de Gerenciamento.

A memória do Processador Central segue um esquema semelhante ao de uma Estação de Trabalho, com a diferença principal que podem existir até 8 bancos de Memória de Escrita e Leitura de 64 KBytes cada um. Cada um desses bancos é constituído de duas páginas de 32 KBytes, sendo possível que se ative qualquer um dos bancos indistintamente. Quando um dos bancos está ativo, qualquer uma de suas duas páginas pode estar ocupando a primeira ou a segunda metade do espaço de endereçamento, porém não é possível ativar simultaneamente páginas de dois bancos diferentes.

As rotinas gravadas na página ROM podem acessar qualquer página de qualquer banco, pois quando a página ROM está ativa, basta selecionar o banco que contém a página desejada, e posicionar a página desejada na segunda metade do espaço de endereçamento. Na Figura 35 está apresentado o circuito de controle da memória do Processador Central.

A página ROM do Processador Central é usada da forma descrita a seguir. No bloco ROM-p estão gravadas as rotinas de conversão de endereços lógicos de setores em discos e disquetes, solicitados pelas estações de trabalho, para endereços físicos de acesso; as rotinas de acesso físico às unidades de disquetes; as rotinas de manipulação da impressora; e as rotinas de acesso ao porto Serial de Comunicação com o terminal de vídeo que pode ser conectado ao Processador Central.

No bloco ROM-0 estão as rotinas de comunicação com as estações de trabalho, as rotinas de gerenciamento de setores de disco da base de dados e rotinas de apoio para as funções de gerenciamento da base de dados. O bloco RAM-e é usado para armazenar os parâmetros de operação e variáveis das rotinas dos blocos ROM-p e ROM-0.

O banco 0 de memória de escrita e leitura é usado para criar um ambiente de programação baseado em CP/M, porém executando as rotinas de gerenciamento da base de dados. Os demais bancos de memória de escrita e leitura são usados para implementar o esquema de gerenciamento de setores da base de dados, e são acessados através das rotinas correspondentes gravadas no bloco ROM-0 da página ROM. Esse esquema de gerenciamento será apresentado com mais detalhe na subseção seguinte.

Figura 35 - Circuito de Controle da Memória RAM do PCG.

A separação da memória de apenas leitura em dois blocos foi feita para permitir que o espaço de endereçamento de 4 a 28 Kbytes, correspondente ao bloco ROM-0, pudesse ser usado por mais de um bloco de memória de apenas leitura, caso os 28 KBytes disponíveis não fossem suficientes para todas as funções necessárias. Nesse caso, os blocos ROM-p e RAM-e seriam mantidos

sempre disponíveis quando a página ROM estivesse ativada, e o bloco ROM-p atuaria como um árbitro para escolher que bloco de 24 KBytes seria colocado no espaço de endereçamento de 4 a 28 Kbytes num dado instante. Esse expediente no entanto não se revelou necessário, pois apenas o bloco ROM-0 supriu toda a memória necessária.

VI.5.3. - O Gerenciamento de Setores da Base de Dados..

O Processador Central usa o Banco 0 de Memória de Escrita e Leitura para manter um ambiente de programação compatível com o Sistema Operacional CP/M. Os demais estão disponíveis para serem usados pelas rotinas que estão gravadas na página ROM. Os demais bancos (que podem atingir até 448 KBytes) são usados para se manter um esquema de "Cache" para as unidades de disco e disquete. Dessa maneira, os setores mais usados podem ser mantidos na memória, e sempre que um acesso é solicitado, o Processador Central verifica se as informações pedidas não estão na memória (ou se podem ser armazenadas temporariamente na memória, no caso de uma escrita).

Sendo duas as atividades do Processador Central, dois são os empregos do Cache para disco. Com o Processador Central operando como um controlador inteligente das unidades de disco e disquete, é possível fazer com que os acessos a setores solicitados pelas Estações de trabalho sejam reconhecidas como fazendo parte de arquivos, acessos a diretório, etc. Pode-se assim manter os diretórios dos discos e disquetes permanentemente na memória, e

sempre que um arquivo é aberto, mesmo que não tenha ainda sido solicitada leitura de setores desse disco, o Processador Central poderia iniciar essa leitura, antecipando a leitura de alguns setores físicos desse arquivo.

Como o Processador Central ativado como um controlador inteligente, depende da disponibilidade do Sistema Operacional adaptado para as condições de Multi-processamento que a máquina implementada impõe, e como essa adaptação não foi feita, o uso da memória cache da maneira descrita acima não foi feito, sendo atualmente o seu uso restrito a manutenção de uma trilha para leitura e uma trilha para escrita de cada unidade de disquete do sistema.

Com o Processador Central operando como um Gerenciador de Bases de Dados apoiado no Modelo de Representação de Objetos, e empregando-se uma estrutura de armazenagem que permita gravar as informações referentes a um mesmo objeto em alguns poucos setores, identificáveis imediatamente a partir da referência ao objeto, pode-se usar a memória cache para manter os setores dos dados dos objetos mais freqüentemente acessados pela Estação de Trabalho que está executando o SACDS, ou para manter os dados necessários aos processos automáticos iniciados a partir de acessos da Estação de Trabalho.

A estrutura de armazenagem empregada deve permitir que se implementem algoritmos eficientes para: que se possa reconhecer rapidamente se um determinado setor está presente na memória cache ou não; estando presente, que se localize rapidamente em que posição da memória cache ele está; que se possa elaborar uma estratégia de escolha de quais setores devem ser mantidos na memória cache, e que permita decidir rapidamente qual setor

substituir se um novo setor deve ser mantido nessa memória e não há mais espaço.

Para efeito do gerenciamento da memória cache, o termo setor da memória cache determina um bloco contínuo de bytes que podem ser lidos ou escritos em um disco com apenas um acesso, porém não necessariamente corresponde a um setor do disco. Para evitar ambiguidade na designação desses dois conceitos, na apresentação que se segue será usado o termo setor apenas para o setor físico do disco, e o termo "Linha" para o setor lógico mantido na memória cache.

Na estrutura desenvolvida, considerou-se que uma linha tem o tamanho de 1 KBytes. Esse número (apesar de poder ser facilmente mudado), foi escolhido por permitir um acesso às trilhas de disquete de maneira que uma mesma linha está sempre contida em uma única trilha, ao mesmo tempo que pode manter um volume satisfatório de dados.

A organização de um disco pelo Sistema Operacional CP/M segue uma estrutura padronizada otimizada para as necessidades desse sistema, porém que não são as mesmas de um SGBD como o que é usado pelo SACDS. Dessa forma, um disquete que seja usado para armazenar uma base de dados não pode conter outras informações que não as da base de dados que armazena, e sua estrutura não é reconhecida pelo padrão CP/M.

A estrutura de dados para o gerenciamento da memória Cache empregado é mostrado na Figura 36 onde o símbolo NP indica a capacidade da memória cache em número de Linhas (Páginas) de 1 KBytes. Como o número de arquivos que mantêm a base de dados são pré-definidos (no caso da implementação atual do SACDS é 5), a identificação de se uma determinada linha está ou não presente na

memória cache é feita pelo arquivo a que ela pertence, usando para isso uma técnica de partição ("Hash") sobre o número do setor do arquivo solicitado. Os setores de um arquivo são trata-

Figura 36 - Estrutura de Dados do Gerenciador da Memória Cache.

dos na tabela de partição "ApontLM" através de agregados de 16 KBytes. Cada posição dessa tabela mantém para cada arquivo a indicação da presença de ao menos um setor do agregado correspondente na memória: se a posição for nula, nenhum setor do agregado está na memória; se for não nula, aponta para um registro da tabela de páginas na memória "LM". Cada um desses registros indica qual linha contém os dados do setor correspondente.

A localização de uma linha na memória é feita verificando-se a existência de ao menos um setor do agregado de 16 KBytes a que o setor solicitado pertence, através da tabela "ApontLM". Se a posição for nula, o setor não estará na memória. Se for não nula, o valor desse índice indica o registro da tabela "LM" onde os setores daquele agregado estão mapeados. Verifica-se a posição correspondente ao número do setor nesse registro. Se a posição

Figura 37 - Exemplo do Uso da Estrutura de Controle da Memória Cache

for nula, o setor não estará na memória. Se for não nulo, o número dessa posição indica o número da linha onde o setor estará armazenado. Verifica-se assim que com apenas duas comparações identifica-se a presença de um setor na memória cache, e localiza-se sua posição.

O algoritmo empregado para a arbitragem de quais setores são mantidos na memória é baseado na quantidade e frequência de acesso a cada setor. Para isso, contam-se o número de acessos que são feitos em cada linha, e armazenam-se as diferenças de contagem de cada linha para linha com o menor número de acessos registrados. Assim por exemplo, se existirem três linhas na memória, que tenham sido acessadas 5, 8 e 10 vezes, estará armazenado os totais 1, 4 e 6. Um novo setor sempre é colocado em uma linha com a contagem de 1 acesso.

As tabelas "LLinhas" e "LTotais" são usadas por esse algoritmo. A tabela "LLinhas" é usada para armazenar uma lista duplamente encadeada de todas as linhas com um mesmo total. As linhas são ordenadas pela ordem de chegada nesse total. Assim, a última linha a ter atingido o total correspondente a uma determinada lista é colocada no final dessa lista. A tabela "LTotais" é usada para armazenar uma lista duplamente encadeada de todos os totais associados a ao menos uma linha. A figura 37 mostra como as listas estariam se existissem duas linhas com total 1, três com total 4 e duas com total 6.

Mantidos dessa forma, o valor absoluto do total não é importante, mas apenas a diferença entre as várias linhas, justamente como o algoritmo necessita. Dessa forma, não é necessário que se

reajustem os totais de acesso conforme as páginas vão sendo acessadas, mas apenas que as listas sejam atualizadas. Cada registro da tabela "LLinhas" está diretamente associado à linha que ela representa (o índice da linha é o mesmo do registro correspondente em "LLinhas". Dessa forma, a cada acesso efetuado em um setor já contido em uma linha corresponde uma remoção de uma célula de uma lista bidirecional (alteração de quatro índices), e a sua inserção em outra lista (alteração de outros 6 índices). Caso o novo total não exista, uma nova célula é inserida na lista armazenada em "LTotais" (alteração de 6 índices). Assim, no pior caso, 16 índices são alterados em cada acesso a uma página já existente na memória. Caso um acesso não se refira a um setor presente na memória, a linha que está em primeiro lugar da lista de menores totais tem seu conteúdo substituído para o setor sendo acessado. Verifica-se assim que a manutenção de dados para a execução do algoritmo de arbitragem dos setores mantidos na memória é também bastante eficiente.

A quantidade de memória utilizada para armazenar todas essas tabelas depende do número máximo de linhas "NL" que são mantidas na memória. Se "NL" for menor ou igual a 255, as tabelas "LM", "LLinhas" e "LTotais" são constituídas de bytes, e ocupam então $24 \times NL$ bytes. A tabela "ApontLM" é também constituída de bytes, e seu tamanho depende do tamanho máximo de cada arquivo, ocupando um byte para cada 16 KBytes de cada arquivo.

VI.6. - Agrupamento das Informações sobre um Mesmo Objeto.

O esquema de acesso à memória cache apresentado na seção anterior, para o uso pelo gerenciador da base de dados, terá sua eficiência aumentada se for possível restringir a dispersão de setores que são acessados, para que a chance de se solicitar o acesso a um setor presente na memória aumente.

Em operação normal, um utilitário do SACDS que se refere a um determinado objeto da descrição, em geral necessita de várias informações sobre esse objeto, informações essas que são solicitadas individualmente para o SGBD. Assim, quando um determinado objeto é solicitado para o SGBD, comumente ocorrerão a seguir várias consultas ou atualizações sobre esse objeto. Se as informações referentes a um mesmo objeto forem mantidas no menor número de setores possível, a dispersão de acesso a setores será então minimizada.

Os arquivos de dados que mantêm uma Base de Dados do SACDS são os seguintes:

- Arquivo de Identificadores: armazena os nomes e sinônimos de cada objeto. Os registros são mantidos em uma estrutura em árvore;
- Arquivo de Objetos: armazena as informações referentes a cada objeto da descrição;
- Arquivo de Relacionamentos: armazena as informações sobre todos os relacionamentos entre objetos existentes na descrição;
- Arquivo de Comentários: armazena os comentários descritos sobre cada objeto da base de dados;
- Arquivo de Descrição de Linguagem: armazena as informa-

ções sobre o esquema de dados da base de dados.

Se não houvesse benefício em se reduzir a dispersão de informações sobre cada objeto, cada um desses arquivos poderia manter as informações espalhadas, em geral na sequência da descrição feita. Havendo esse benefício, as informações devem ser agrupadas, e a forma de fazê-lo varia de arquivo para arquivo.

O arquivo de identificadores é usado para localizar um objeto por um seu identificador. Como o arquivo é estruturado em árvore binária, é interessante que cada setor de 1 KByte contenha apenas uma sub-árvore. Assim, a raiz de uma árvore estará em um setor juntamente com os identificadores que lhe são adjacentes na árvore. Quando um ponteiro da árvore apontar para um identificador armazenado em outro setor, faz-se com que esse novo setor armazene apenas os identificadores que são subordinados a aquele identificador.

O arquivo de objetos não necessita de nenhuma re-estruturação especial, visto que não existe nenhuma lei que permita prever qual o próximo objeto que será interessante acessar depois do acesso a qualquer um determinado objeto.

O arquivo de relacionamentos deve ser estruturado de maneira que os relacionamentos referentes a cada objeto sejam mantidos agrupados. Como não se pode prever o número de relacionamentos que um objeto terá, e nem se pode considerar que os relacionamentos de um determinado objeto serão todos descritos simultaneamente, então o que se pode fazer é prever o espaço para bloco de relacionamentos de tamanho pré-fixado. Sempre que um novo relacionamento de um objeto for colocado, ele deve ser colocado num bloco já alocado para os relacionamentos daquele objeto. Somente

se não houver mais espaço um novo bloco será alocado, e este estará reservado para relacionamentos do mesmo objeto. Como um registro do arquivo de relacionamento do SACDS ocupa 17 bytes, cabem 60 registros desse tipo em um setor de 1 KBytes. Escolheu-se então que cada bloco de registros de relacionamentos teria espaço para armazenar 20 relacionamentos.

Note-se que com essa técnica, passam a existir muito mais "espaços vazios" em um arquivo de relacionamentos, devido aos blocos alocados para relacionamentos de cada objeto que não estão totalmente preenchidos. Uma forma de se otimizar o par de necessidades concorrentes aumento-do-arquivo/dispersão-aceitável consiste em se variar o tamanho dos blocos de relacionamentos. A escolha desse parâmetro depende da velocidade de acesso que se pretende, e do espaço em disco que se permite ocupar.

O arquivo de comentários deve ser estruturado da mesma maneira que o arquivo de relacionamentos. Porém como comentários normalmente correspondem a textos, que podem ser extensos, optou-se por manter cada bloco de comentários armazenado em um setor do arquivo de comentário.

O Arquivo de Definição de Linguagem, quando em uso normal pelo SGBD, é usado apenas para leitura, com acesso seqüencial a blocos de informação. Esses acessos, além de esporádicos, são totalmente aleatórios, do ponto de vista do gerenciamento de memória cache. Por isso, nenhuma estruturação adicional lhe foi imposta.

VI.7. - Conclusões.

Neste capítulo foram apresentados a implementação de uma máquina para dar suporte ao SACDS, e os conceitos e considerações usados para o seu desenvolvimento. A apresentação feita relata a Arquitetura da máquina tal como está implementada, a qual corresponde aos objetivos iniciais de servir de suporte à execução do SACDS. Com o delineamento do Modelo de Representação de Objetos, uma nova arquitetura, específica para as características do MRO, passou a ser desejável.

Essa arquitetura pode ser vista como uma extensão àquela já implementada, da mesma forma como o MRO pode ser visto como uma extensão ao Modelo de Dados do SACDS. Decorrentes do desenvolvimento da Máquina SACDS/SGBD, e das avaliações feitas com ele, chegou-se à proposta de uma nova arquitetura, mais adequada para o atendimento das necessidades de um Sistema de Gerenciamento de Dados para Engenharia. As considerações para essa proposta serão apresentadas no capítulo seguinte.

C A P Í T U L O V I I

CONSIDERAÇÕES SOBRE UMA MÁQUINA DE BASE DE DADOS PARA O MODELO

DE REPRESENTAÇÃO DE OBJETOS

VII.1. - Introdução.

A máquina (SACDS/SGBD) apresentada no capítulo anterior apresenta alguns aspectos que devem ser analisados. Em primeiro lugar, como os cinco processadores apresentam um ambiente para a execução de programas em CP/M, e a memória de cada processador não é diretamente acessada por nenhum outro processador, essa máquina pode ser considerada como uma rede de processamento distribuído. Por outro lado, como o Processador Central realiza todas as operações de gerenciamento da base de dados, e as Estações de Trabalho executam os programas de aplicação que utilizam essa base de dados, então o processador central pode ser considerado, a grosso modo, como uma Máquina de Base de Dados ("Database Machine"). Chega-se então à conclusão de que essa máquina constitui-se efetivamente de um sistema distribuído, que incorpora como um de seus nós, uma máquina de base de dados, à semelhança da máquina DELTA [TAK_85].

A motivação para o desenvolvimento dessa máquina era a de

tirar do processador que estivesse executando uma ferramenta do SACDS, as tarefas de gerenciamento da base de dados.

Decorrentes do desenvolvimento e das avaliações feitas com a Máquina SACDS/SGBD, chegou-se à proposta de uma nova arquitetura, mais adequada para o atendimento das necessidades de um Sistema de Gerenciamento de Dados para Engenharia. Essa proposta concentra-se na expansão do Processador central para tornar-se uma Máquina de Bases de Dados capaz de atender a várias Estações de Trabalho, cada uma delas podendo executar programas de aplicação que usem essa máquina, sem a restrição de apenas uma poder fazê-lo ao mesmo tempo. Dessa forma, pode-se imaginar o novo sistema como um conjunto de Estações de Trabalho acessando a Máquina de Base de Dados concorrentemente.

Restringindo a atuação dessa máquina para ser apenas um gerenciador de bases de dados, as tarefas de controlar as unidades de disco e disquete não envolvidas com a base de dados em si deixam de ser atribuição sua. Essas tarefas devem passar a ser desempenhadas por outro componente do sistema. Coloca-se então três alternativas para a maneira como a Máquina de Base de Dados em si comunique-se com os programas de aplicação que a utilizam:

a) fazer com que cada Estação de Trabalho tenha acesso aos seus próprios discos. Essa alternativa mantém a caracterização de que cada estação de trabalho é um sub-sistema que, a menos dos acessos à base de dados, pode operar totalmente independente das demais.

b) fazer com que cada estação de trabalho tenha acesso tanto à Máquina de Base de Dados quanto a um segundo subsistema que cuidaria especificamente dos dispositivos de disco, disquete, impressoras, e demais periféricos centralizados.

Essa alternativa mantém a independência das estações de trabalho, e aumenta a utilização desses dispositivos centralizados, porém acarreta maior complexidade no elo de comunicação entre Estações de Trabalho, Máquina de Base de Dados e Sub-Sistema de Discos.

c) fazer com que todas as estações de trabalho tenham acesso à Máquina de Base de Dados através de um computador de grande porte (Computador hospedeiro), fazendo com que cada Estação de Trabalho seja na realidade apenas um terminal desse computador.

A alternativa c) é a mais comumente adotada pelas demais Máquinas de Base de Dados existentes, que em geral são ligadas a um processador hospedeiro que "compila" todos os pedidos de acesso a bases de dados dos programas de aplicação, e efetua as consultas correspondentes para a Máquina de Base de Dados. Alguns exemplos disso são as máquinas DIRECT [WIT_79], HAS [HAH_85] e RAP.2 [SCH_79]. As alternativas a) e b) podem ser usadas quando as estações de trabalho estão conectadas através de uma rede local. Essa forma de comunicação é mais lenta do que a taxa de transferência que a Máquina de Base de Dados e as estações de trabalho poderiam suportar, porém atualmente corresponde à alternativa técnica que tem sido empregada em sistemas desse tipo, tal como é o projeto japonês de sistemas de quinta geração.

A seguir, será feita a apresentação de uma proposta para uma base de dados, concebida a partir do desenvolvimento do sistema descrito no capítulo VI, voltada para apoiar as tarefas de processamento de bases de dados do MRD. De acordo com as classificações propostas por [QAD_85], essa máquina teria arqui-

tetura MIMD, com processamento em memória dedicada, e indexação a nível de páginas.

Quanto à maneira como essa máquina se comunica com os programas de aplicação, corresponde àquela em que cada estação de trabalho se comunica diretamente com a Máquina de Base de Dados, uma vez que não é necessário a existência de um processador hospedeiro compilando os pedidos de acesso gerados pelas estações de trabalho.

VII.2. - A arquitetura da Máquina de Bases de Dados Proposta.

A Máquina de Bases de Dados proposta apresenta uma arquitetura não convencional entre as Máquinas de Bases de Dados existentes, no sentido que apresenta-se como uma máquina dedicada ao processamento de bases de dados de um modelo não convencional. Além disso, dado que as necessidades impostas por esse modelo são em larga escala distintas das impostas pelos modelos convencionais, a arquitetura resultante acaba se tornando diferente das que normalmente são empregadas em máquinas dedicadas a tarefas de bases de dados usando modelos convencionais. Por outro lado, a arquitetura proposta se aproxima das arquiteturas existentes de máquinas de bases de dados que empregam conceitos mais inovativos, tal como a máquina SiDBM [LEL_85], que não usa discos, mas mantém toda a informação em memória principal.

A principal diferença que deve ser notada entre as Máquinas de Bases de Dados existentes e a aqui sugerida, é que os modelos convencionais necessitam pesadamente de operações de procura de

registros baseados no conteúdo desses registros, tal como é necessário nas operações de seleção e união no modelo relacional, ao passo que no Modelo de Representação de Objetos essa necessidade é bastante reduzida. Outra diferença é que operações de ordenação no MRO são necessárias apenas quando da colocação de novos objetos, e mesmo assim, deve levar-se em conta que cada objeto é colocado através de operações solicitadas individualmente para a base de dados. Nas Máquinas de Bases de Dados em geral existem módulos de "Hardware" destinados especificamente para ordenação de dados, porém na Máquina sugerida para o MRO isso não é feito.

Figura 37 - Arquitetura da Máquina Proposta.

A Figura 37 mostra a arquitetura para a máquina proposta. Pode-se notar que ela é composta por quatro tipos de processadores:

PL - Processadores de Ligação

PM - Processadores de Manipulação de Dados;

PP - Processador de Pedidos de acesso;

CD - Controlador das Unidades de Disco.

Além disso, existem dois dutos internos que interligam os vários processadores: Duto de Dados e Duto de Páginas. Ambos são dutos Sincronos paralelos, de alta velocidade. O Duto de Dados é usado para enviar aos Processadores de Manipulação (PM) os pedidos de acesso solicitados pelas estações de trabalho, e para a troca de dados entre as estações de trabalho e os Processadores de Manipulação. O Duto de Páginas é usado para carregar as memórias locais dos Processadores de Manipulação com os dados necessários à execução das solicitações de acesso que cada um estiver atendendo, bem como para a gravação em disco das páginas modificadas por essa execução.

Os Processadores de Ligação são usados para atuar como interface entre cada Estação de Trabalho e o Duto de Dados. Não efetuam nenhum processamento lógico dos dados que manipulam, a menos da substituição ou preenchimento de alguns campos nos pedidos de acesso que manipulam. Atuam principalmente como "buffers" e temporizadores para os dados que trafegam pelo Duto de Dados e que têm como origem e destino aquela específica Estação de Trabalho. Os pedidos de acesso de cada Estação de Trabalho consistem de conjuntos de parâmetros e dados que são introduzidos (ou recebidos) pelo Processador de Ligação no Duto de Dados como

um "Pacote de Dados". Esse processador coloca em cada pacote de dados uma identificação de qual é o Processador de Ligação que originou o pacote de dados, e um sinal indicando se o pacote atual corresponde a alguma operação já iniciada anteriormente, e nesse caso, qual o Processador de Manipulação que a havia atendido.

Cada pacote originado em uma Estação de Trabalho, uma vez tendo sido lançado no Duto de Dados, é "ouvido" por todos os Processadores de Manipulação, que verificam se esse pacote deve ser tratado por ele, ou se não for indicado nenhum processador específico, se o pacote pode ser atendido pelos dados presentes na memória do Processador. Quando um Processador pode atender ao pedido, ele envia ao final do pacote de dados um sinal indicando a aceitação do pacote.

O Processador de Manipulação consulta os dados que tem disponíveis, e procura atender a todos os pedidos que tenha aceitado, em geral na ordem de chegada dos pedidos. Um pedido pode ocasionar três tipos de resposta do Processador de Manipulação que o estiver atendendo:

- finalização do pedido. Nesse caso o Processador coloca um novo pacote de dados no Duto de Dados, informando como destino o Processador de Ligação que originou o pedido. Nesse pacote estão a indicação de estado de finalização do pedido, e dados que possam ter sido solicitados.
- solicitação de uma tarefa auxiliar em algum outro processador. Essas tarefas são destinadas a realizar algum processamento acionado automaticamente a partir de outras tarefas em execução. Não são prioritárias para a execução do pedido em andamento, e por isso são armaze-

nadas para execução quando houver processador desocupado. Nesse caso o Processador de Manipulação coloca um novo pacote de dados no duto, que mantém as mesmas características dos que são originados pelos Processadores de Ligação, informando os dados e parâmetros necessários, porém com a informação de que não é um pedido prioritário.

- Solicitação de dados não disponíveis para a execução de uma tarefa em andamento. Nesse caso o Processador de Manipulação coloca um pedido de dados no Duto de Dados, e passa o pedido em andamento para o final da fila de pedidos pendentes, aguardando a disponibilidade dos dados solicitados.

O Processador de Pedidos monitora todos os pacotes que passam no Duto de Dados, e verifica os sinais de aceitação com que os Processadores de Manipulação respondem aos pacotes que podem atender. Dessa forma, o Processador de Pedidos pode manter um registro de todas as operações que estão sendo atendidas em cada Processador de Manipulação. Quando nenhum Processador de Manipulação envia o sinal de aceitação para um pacote, esse processador verifica qual processador tem espaço de memória com capacidade para receber os dados inicialmente necessários para o pedido, que esteja há mais tempo sem uso, e que esteja com baixa carga de trabalho. Localizado um Processador de Manipulação nessas condições, o Processador de Pedidos solicita o carregamento dos dados adequados ao Controlador das Unidades de Disco, que efetua a transferência dos dados armazenados em uma unidade de disco na memória do Processador de Manipulação

escolhido. Ao final dessa transferência, o Processador de Pedidos coloca novamente no Duto de Dados o pacote que não havia sido aceito, tendo como endereço de destino explícito o Processador cuja memória recebeu os dados adequados.

O Processador de Pedidos cuida também dos Pacotes enviados pelos Processadores de Manipulação. Os pedidos desses processadores solicitando mais dados são tratados de forma semelhante ao anterior, com a diferença que a indicação para o Controlador das Unidades de Disco de qual é o processador destino de uma transferência já vem especificada. Além disso, ao final dessa transferência o Processador de Pedidos coloca no Duto de Dados apenas uma indicação de que os dados pedidos já podem ser acessados.

Os pacotes informando a necessidade de execução de tarefas automáticas, que são colocadas no Duto de Dados pelos Processadores de Manipulação, são recolhidas pelo Processador de Pedidos, e armazenadas em memória própria. Quando este detecta que existe um Processador de Manipulação sem nenhum pedido pendente, e com espaço de memória que não está sendo utilizado por um longo período de tempo, então ele escolhe um dos pedidos de execução armazenados, e processa o pacote correspondente como se ele fosse um pedido normal que não tenha sido aceito por nenhum processador, ou seja, torna disponível ao Processador inativo os dados necessários, e lança novamente aquele pacote no Duto de Dados.

O Controlador de Unidades de Disco é apenas um processador capaz de fazer a transformação de pedidos de dados lógicos, que são manipulados por todos os demais processadores, em endereços físicos em disco, identificando os discos e trilhas que devem ser

acessados. Esse processador poderá também fazer a busca escolhendo caminhos de acesso mais curtos, no caso de haver uma sucessão de pedidos de acesso.

Deve-se ressaltar que o Processador de Pedidos, os Processadores de Manipulação e o Controlador das Unidades de Disco utilizam processadores cujos programas são em parte permanentemente residentes, tais como aqueles que permitem a inicialização do sistema e o carregamento de outros programas, e outros são carregados de disco usando o mesmo duto de páginas usado para o carregamento dos dados da Base de Dados.

VII.3. - A Arquitetura de um Processador de Manipulação.

A arquitetura de um Processador de Manipulação e a do Processador de Pedidos são em geral semelhantes, e serão tratadas nessa seção como se fossem idênticas. No final serão apresentadas as diferenças que devem existir para que um processador possa ser usado como um Processador de Pedidos. Um ponto que deve ser colocado de início é que os Processadores de Manipulação não são necessariamente todos iguais, podem porém especializar-se em alguns tipos, para executar com mais eficiência determinados tipos de processamento necessários numa base de Dados que utilize o MRO. As diferenças no entanto, consistem apenas em dimensionamentos de memória e de alguns circuitos auxiliares.

Outro ponto importante é que a Unidade Central de Processamento (Micro-processador) utilizada em cada Processador não é crítica, podendo mesmo existir Processadores que usam

diferentes Micro-processadores num mesmo sistema. O problema causado pela existência de vários Micro-processadores diferentes seria apenas a necessidade de se manter cópias dos mesmos programas mantidos para as diferentes linguagens de máquina de cada Micro-processador. Esse ponto é importante em um sistema com a complexidade da Máquina de Base de Dados aqui proposta, pois permite que seu desenvolvimento possa acompanhar por muito tempo o rápido desenvolvimento que hoje existe nessa área.

Figura 38 - Arquitetura de um Processador de Manipulação.

A Figura 38 mostra a arquitetura geral de um Processador de Manipulação, onde pode-se notar que contém todos os elementos para se constituir em um computador convencional independente, otimizado para operar gerando e recebendo dados dos dois dutos com que se interliga. Essencialmente, consta de uma unidade Central de Processamento (UCP), dois ou mais bancos de memória com possibilidade de operação de transferência de dados através de Acesso Direto à Memória, e interface para o Duto de Dados.

A memória deve ser dividida em bancos, e o registro de qual é o conteúdo de cada banco deve poder ser conhecido tanto pela UCP como pela interface para o Duto de Dados. Além disso, o Processador de Pedidos pode travar o acesso a alguns blocos dessa memória para poder autorizar a transferência de dados ao Controlador de Unidades de Disco. A estruturação em bancos de memória facilita a operação de travamento, fazendo com que o bloco travado corresponda a um banco.

Os pedidos de acesso de dados que se originam nas Estações de Trabalho se referenciam a objetos através de seus códigos (o parâmetro Código de Objeto utilizado nas rotinas apresentadas no Capítulo IV). Esses códigos são usados pela Máquina de Base de Dados para derivar o índice da página de dados do arquivo de Objetos da base de dados. Essas páginas são as mesmas que foram apresentadas na seção VI.6 com respeito ao Arquivo de Objetos, com a diferença que nessa máquina o tamanho dessas páginas deve ser maior, tal como será tratado na seção VII.7. O tamanho a ser utilizado depende de experimentação ou simulação, e não se poderia agora estabelecer esse valor com segurança, porém considere-se que pudesse ser 8 KBytes, e que o tamanho de um banco de memória pudesse ser 64 KBytes. Nesse caso cada banco de memória poderia conter 8 páginas. Tal como foi visto na seção VI.6, um certo número pré-fixado de registros de objetos poderia estar contido em uma página. Usando, apenas como exemplo, o número de bytes atualmente usados por um registro do arquivo de objetos do SACDS, que é 40, poderiam ser armazenados até 204 objetos em uma página de 8 KBytes.

O número índice de cada página armazenada em um banco de dados do Processador é o dado conhecido pela interface do Duto de

Dados. Nesse caso, sempre que houvesse uma referência a um objeto, a própria interface poderia verificar através de seu código se o processador pode atender ao pedido ou não, e ao final de um pacote de dados recebido ela poderia imediatamente enviar o sinal de aceitação do pedido.

E' atribuição do Processador de Pedidos zelar para que não exista a mesma página de dados duplicada na memória de mais de um processador ao mesmo tempo, e assim, uma vez que um processador aceite um pedido, nenhum outro o terá feito. Esse fato, além de simplificar o controle do sinal de aceitação do pedido, garante a consistência das alterações efetuadas, eliminando a necessidade de se efetuar travas de consistência sobre os registros ou páginas que devem ser modificados.

A interface do duto de dados é um circuito específico, capaz de comparar campos pré-determinados de um pacote de dados com os índices de todas as páginas não travadas presentes na memória do processador. Além disso, deve ter um "buffer" capaz de armazenar até dois pacotes de dados inteiros, antes de, através da aceitação de um pedido, passá-lo para a memória principal do processador.

O Processador executa os pedidos aceitos pela interface do Duto de Dados seqüencialmente, e sempre que estiver usando um banco de dados, deve travá-lo para evitar o seu travamento pelo Processador de Pedidos. Além disso, sempre que recebe um pacote da interface, analisa os índices das páginas implicadas por esse pedido, e mesmo que não o processe imediatamente, indica que a página correspondente não deve ser retirada de sua memória. O número de pedidos de acesso pendentes que necessitam de cada página disponível deve ser contado, apesar de apenas a informação

de estar ela prendendo uma página ser informado ao Processador de Pedidos. Quando o final de processamento de um pedido resultar em não mais serem necessários os dados de uma determinada página para nenhum dos pedidos pendentes, essa informação torna-se disponível ao Processador de Pedidos, que a utiliza para avaliar se esse processador pode ter essa página substituída de modo a poder atender a um pacote de dados enviado por uma Estação de Trabalho que não foi aceito por nenhum outro Processador de Manipulação.

O processador de pedidos pode verificar quais são as páginas que não podem ser alteradas em cada banco de memória, através de uma leitura direta pela linha de acesso do Processador de Pedidos. Além disso, pode da mesma maneira verificar se pode travar um bloco de memória para a substituição de uma de suas páginas ou não. Uma página poderá ser substituída quando não houver indicação que um pedido pendente esteja travando a página, e um banco pode ser travado quando o pedido sendo processado pelo Processador de Manipulação não estiver usando aquele banco.

VII.4. - O Formato de um Pacote de Dados.

Todos os pacotes de Dados colocados no Duto de Dados têm o mesmo formato, independente de qual tenha sido a sua origem. A transferência de um Pacote de Dados é efetuada através da requisição de controle do duto de dados por um processador. Uma vez aceito o pedido, o processador que solicitou a requisição passa a atuar como mestre do duto, e todos os demais

processadores passam a atuar como escravos, interpretando o pacote.

O pacote será então enviado, através da transferência byte a byte em paralelo de todo o pacote, sincronizados pelo controlador do duto.

Figura 39 - Formato de um Pacote de Dados.

O formato de cada pacote é mostrado na Figura 39. O primeiro byte indica o código de grupo da função solicitada por esse pacote. Cada grupo de funções corresponde a funções que somente podem ser desempenhadas por processadores específicos. Por exemplo, uma solicitação de obter dados de um objeto não pode ser desempenhada por um Processador de Ligação, e tampouco o pacote de dados que responde a essa solicitação pode ser recebido por um Processador de Manipulação. Assim, o Código de Grupo de Função serve para desativar as Interfaces que estão conectadas a Processadores que não podem responder a esse Pacote.

Seguindo ao código de grupo vem o código da própria função solicitada. A seguir vem a informação sobre o Processador Destino do pacote. Esse campo é usado quando a mensagem tem um

Processador Destino já especificado. Nesse caso o processador destino (a menos que tenha sido desativado pelo código de grupo da função) deve obrigatoriamente aceitar o pacote de dados.

A quarta informação enviada corresponde ao índice da página que é necessariamente referenciada por essa solicitação, tal como o código do objeto se a solicitação corresponder a um pedido de dados sobre um objeto. Esse campo será discutido com maior detalhe na seção VII.6.

A seguir vem a informação de qual é o processador origem desse pacote, para que o Processador que aceitar o pacote possa designá-lo diretamente em um eventual pacote de resposta. Essa informação não é modificada pelo Processador de Pedidos se ele for retransmitir essa mensagem, permanecendo a informação do processador origem que primeiro transmitiu o pacote.

Depois do oitavo byte, cada pacote contém um número variável de bytes que contêm os parâmetros e dados da função solicitada. Esse número deve ter um valor mínimo e máximo a ser determinado. A princípio, um número mínimo de 4 bytes deve ser considerado, para permitir a inicialização do processador que aceitar o pacote, e um número máximo de 120 bytes seria suficiente para os dados necessários para qualquer das funções do Modelo de Representação de Objetos.

O final de um pacote de dados deve ser sempre seguido por um sinal de aceitação de um processador que tenha recolhido e aceito o pacote. Esse final é indicado pelo processador origem da mensagem através de um sinal de controle do duto de dados, e informa a todos os demais processadores o final desse pacote, com a indicação de liberação do duto. O sinal de aceitação de um pacote é dado usando um período a mais do duto, para a

transmissão do sinal de aceitação do processador destino, o qual é acompanhado de um Byte contendo o número do processador que o enviou. Para a transmissão desse byte, não é necessária a solicitação de uso do duto, uma vez que somente um processador terá condições de enviar esse sinal.

VII.5. - A Interface do Duto de Dados.

A arquitetura de cada Processador que compõe a Máquina proposta usa em geral componentes convencionais, a menos da Interface para o Duto de Dados, que deve dispôr de um "hardware" especial. Em todos os processadores dessa máquina, existe uma Interface para o Duto de Dados. A constituição dessa interface em cada tipo de processador é diferente, dependendo das funções necessárias em cada processador. Ela se apresenta com o maior número de funções necessárias no Processador de Manipulação, onde deve ser capaz de:

a) efetuar a comparação de campos específicos de cada pacote de dados com o conteúdo de vários registradores de índices de páginas ao mesmo tempo. Essa comparação deve envolver todos os índices das páginas presentes em todos os bancos de memória não travados do processador.

b) armazenar até dois pacotes de dados ao mesmo tempo. Uma vez aceito um pacote e enviado o sinal de aceitação correspondente, o pacote seguinte já pode ocupar o Duto de Dados e a Interface já deve estar pronta para recebê-lo. Ao mesmo tempo, o primeiro pacote, já aceito, deve ser levado para a memória principal do

Processador.

c) requisitar o uso do Duto de Dados, e transmitir por esse duto um pacote de dados.

A primeira função pode ser executada através de comparadores associados a cada registrador que armazene um índice de página. Essa função seria implementada mais facilmente se os registradores de página forem registradores de deslocamento, e houver tempo para que entre cada Byte transferido pelo duto de dados sejam deslocados e comparados oito bits em cada registrador de índice.

A segunda função pode ser executada se cada pacote de dados for armazenado em uma memória com estrutura de fila ("FIFO"), e o processador for rápido o suficiente para ler dessa memória um byte na mesma velocidade com que os bytes são transmitidos no duto de dados. Caso contrário, capacidade para mais pacotes de dados, e uma estrutura de acesso direto à memória será necessário. Um tempo de resposta igual entre a leitura de um byte pelo processador e a transferência de um byte do duto de dados é uma condição suficiente para que o próprio processador possa efetuar a cópia da memória da interface para a memória principal, uma vez que o sinal de aceitação de um pacote somente é efetuado ao final da transmissão, de todo o pacote, porém é detectado pela interface logo ao final da transmissão do índice de página do pedido. Como depois da transmissão desse índice o pacote ainda continua com a transmissão de mais dados, existe tempo hábil para que a interface interrompa o programa em execução no processador principal e permite que este se prepare para o início da cópia desse pacote logo ao final de sua transmissão.

A terceira função da interface pode ser desempenhada tanto

lendo o pacote diretamente do processador (se a condição de tempo permitir), e nesse caso não é necessária a existência de outra memória de Fila na interface, ou então (se aquela condição não permitir) deve existir outra memória de fila na interface dedicada a essa função.

Os processadores que não os de manipulação podem ter uma interface mais simples, uma vez que não precisam desempenhar algumas dessas funções. Assim, o Processador de Pedidos não precisa comparar os índices das páginas para confrontar com as suas próprias, mas deve verificar apenas o código de grupo de função de cada pacote de dados. Os Processadores de Ligação por sua vez devem verificar apenas o Processador destino de cada pacote, e enviar o pacote à sua estação de trabalho apenas quando o processador destino indicar esse processador.

O Controlador do Duto de Dados desempenha todas as tarefas de temporização do duto de dados, atribuição de processador mestre para cada transferência e sinalização de controle do duto. No entanto, para que todos os pacotes transmitidos possam ser verificados pelo Processador de Pedidos, este último atua como um fiscal do Duto, indicando ordem de aceitação de requisições de uso do duto, por tipo de processador, e liberando ou restringindo o uso do duto.

VII.6. - A Identificação de Páginas que Contêm as Informações Pedidas.

Na seção VI.5 foi apresentada a forma como um arquivo de dados da Base de Dados do SACDS agrupa os itens de dados que potencialmente são acessados juntos. A mesma maneira de agrupar esses itens deve ser usada pela Máquina proposta, porém levando em conta as necessidades do Modelo de Representação de Objetos e as características da própria máquina.

Será feita aqui uma consideração de como os arquivos de dados de uma Base de Dados que usa o MRO podem ser constituídos, em termos de como pode ser feito o agrupamento de dados em cada um desses arquivos, se o tamanho das páginas da Máquina proposta forem aqueles sugeridos na seção VII.3, ou seja, páginas de 8 KBytes.

Além dos Arquivos de Objetos, de Identificadores, e de Relacionamentos que existem na Base de Dados do SACDS, uma base de dados usando o Modelo de Representação de Objetos irá necessitar dos Arquivos de Classes, de Atributos, e além do Arquivo de Comentários, de arquivos de todas as Características de Atributos que a particular implementação suportar.

Todos esses arquivos devem manter uma estrutura que permitam identificar com facilidade em qual página na Base de Dados a informação está. Nas seções anteriores foram feitas considerações sobre como a Máquina de Base de Dados proposta pode fazer a identificação da página onde se encontram as informações sobre um determinado objeto no Arquivo de Objetos, usando para isso o código do objeto. Essa forma de identificação é exclusiva do acesso a objetos, uma vez que o código de cada objeto pode ser

conhecido, e portanto indicado pelo próprio programa de aplicação em execução nas Estações de Trabalho, ao passo que as demais entidades da base de dados não têm seu código conhecido fora da própria Base de Dados. No entanto, a referência a cada entidade continua internamente a ser feita através de códigos, que são os índices de cada entidade no seu próprio arquivo. Assim, por exemplo, cada Identificador tem seu próprio código, bem como cada linha de comentário, apesar de esses códigos não passarem para o programa de aplicação.

Verificando como as rotinas de acesso à Base de Dados foram especificadas no Capítulo IV, verifica-se que o acesso a cada uma das entidades da base de dados que não os objetos, é feita posicionando-se nessas entidades através das associações de cada entidade aos objetos, que esses sim são indicados explicitamente. O posicionamento faz com que exista ao mesmo tempo uma entidade de cada tipo posicionada de cada vez, que é denominada a entidade corrente. Assim, através de posicionamentos, pode-se ter um relacionamento corrente, um identificador corrente, uma linha de comentário corrente, etc.

O reconhecimento interno de qual é cada entidade corrente é feita armazenando-se o código de cada entidade que é considerada corrente num dado instante. Quando um pedido de acesso refere-se a uma entidade corrente, digamos do relacionamento corrente, a informação de que se trata do relacionamento corrente é substituída pelo código do relacionamento corrente, e a partir daí tudo ocorre como se o acesso fosse feito explicitamente ao relacionamento indicado. Assim, para que a localização de uma página de qualquer arquivo possa ser feita, basta usar o código do elemento sendo acessado para obter o índice da página que

contém essas informações.

Para que um pacote de dados que se refere a qualquer entidade possa ser colocado no duto de dados da máquina proposta, é necessário que o código da entidade seja colocado no pacote explicitamente. Como em um dado instante existe somente uma entidade corrente para cada tipo de entidade em cada estação de trabalho, então a identificação de qual é o conjunto de entidades correntes para cada estação de trabalho deve ser feita pelo Processador de Ligação. Dessa forma, sempre que um pedido de acesso a qualquer entidade seja feito por uma Estação de Trabalho, o seu respectivo Processador de Ligação completa o pedido, adicionando-lhe a identificação da entidade envolvida, e só então coloca o pedido como um pacote no Duto de Dados.

Assim o tratamento de todos os tipos de entidades manipulados pela base de dados tem um tratamento homogêneo, seja a entidade um objeto, um relacionamento, ou qualquer outro tipo de entidade.

O conjunto de operações para executar a marcação de estado apresentado na seção IV.4.1.1., deve ser então executada através da parceria entre o Processador de Ligação que atende a uma Estação de Trabalho e os Processadores de Manipulação que atendem a cada uma de suas solicitações. Essa parceria pode distribuir o processamento necessário de várias formas diferentes, indo desde a manutenção de toda a estrutura de dados própria para marcação de estado (Pilha de Estados, Campos de Usuário, Registro de Entidades Correntes, etc.) no próprio Processador de Ligação, até a manutenção nesse processador apenas do Registro de Entidades Correntes.

Os códigos de todas as entidades que circulam pelo Duto de Dados vão servir para que cada Processador de Manipulação verifique se pode atender cada pedido ou não. Como isso é feito por um circuito específico de reconhecimento, a forma de reconhecimento deve ser única para qualquer tipo de entidade. A nível de circuito, o ponto principal a ser considerado é o número de bits que deve ser comparado. Considerando que sejam usados 32 bits (4 bytes) para compôr cada código de entidade, o número de bits a serem comparado depende da quantidade de registros de cada tipo de entidade que é colocado em uma só página.

Tomando apenas como exemplo a Base de Dados do SACDS, seriam necessários 40 Bytes para cada registro de objeto, 65 para cada registro de identificador, 85 para cada registro de comentário e 17 bytes para cada registro de Relacionamento. Se o tamanho de cada página fosse o proposto na seção 6.5, de 8 KBytes, então a tabela seguinte mostra o número máximo de registros que poderiam ser armazenados em cada página.

Tipo da Entidade	Tamanho do Registro (Bytes)	Registros por Página	Número de Códigos	Nº de Bits Comparados
Objeto	40	204	256	24
Identificador	65	126	128	25
Relacionamento	17	481	512	23
Comentário	85	96	128	25

Para que cada índice da página possa ser comparado diretamente com o código de cada entidade, será necessário que a quantidade de códigos que estejam contidos em cada página seja um número múltiplo de uma potência de dois. Assim por exemplo, os objetos de código 1 a 204 estarão na primeira página do arquivo

de objetos, os números 205 a 255 não serão código de nenhum objeto, os objetos de código 256 a 459 estarão na página dois desse arquivo, e assim por diante. Então, dos 32 bits que compõem cada código, apenas 24 precisarão ser comparados para se identificar a página onde o objeto está. Da mesma forma, a tabela acima mostra o número de bits que devem ser comparados para a identificação da página de cada uma das entidades que compõem aquela Base de Dados.

Uma Base de Dados usando o MRO terá além desses quatro, os outros arquivos que compõem uma Base de Dados desse modelo. Além disso, os registros que compõem cada um dos arquivos que já existem na Base de Dados do SACDS serão diferentes na Base de Dados que usa o MRO, sendo todos eles maiores (em número de bytes) do que os da Base de Dados do SACDS, porém nenhum deles deverá ultrapassar um tamanho em torno de 200 bytes, o que corresponde à necessidade de se comparar entre 22 e 25 bytes do código de uma entidade para se identificar qual página armazena aquela entidade.

Existem duas possibilidades para se fazer o circuito que realiza essa comparação: fazer com que o número de bits comparados seja efetivamente o necessário para cada comparação, ou fazer o tamanho das páginas diferentes para cada tipo de entidade. Embora as duas alternativas devam ser consideradas, a primeira parece ser melhor, devido ao fato de que afeta apenas o circuito da Interface do Duto de Dados de cada Processador de Manipulação, ao passo que a segunda afeta o número de registros que armazenam os índices das páginas que estão na memória de cada processador, o que acarreta um algoritmo que leve em conta essas diferenças até ao nível do Processador de Pedidos.

VII.7. - Os Vários Tipos de Processadores de Manipulação.

Para que o esquema de identificação de páginas proposto na seção anterior possa ser eficiente, é necessário que as informações referentes a um mesmo objeto sejam mantidas agrupadas. Assim, o mesmo critério para o agrupamento de dados empregado na Máquina SACDS/SGBD deve ser adotado na Máquina agora proposta. Para manter-se uma visão geral de como esse agrupamento deve ser feito, será feita aqui uma apresentação de como devem ser agrupadas as informações em cada um dos arquivos da Base de Dados usando o MRO.

O Arquivo de Definição de Linguagem do SACDS corresponde em uma Base de Dados usando o MRO à Meta-Base de Dados, que contém a indicação da estrutura das informações contidas na Base de Dados em si. Como na operação normal da base de dados essa Meta-Base de Dados é usada apenas para leitura, ou então tratada como um Base de Dados igual a qualquer outra (para atualização), então não existe um arquivo que deva ser tratado de forma especial. Os demais arquivos de uma Base de Dados usando o MRO, com a respectiva maneira como devem ter as informações agrupadas, são os seguintes:

-- Arquivo de Classes: armazena os ponteiros de acesso a cada conjunto de identificadores das classes constritas pelo objeto que abre cada classe (raízes das árvores). Não necessita de nenhuma re-estruturação especial, visto que não existe nenhuma lei que permita prever qual a próxima classe que deverá ser acessada após um acesso a qualquer classe.

-- Arquivo de Identificadores: armazena os nomes e sinônimos de cada objeto. Os registros são mantidos em uma estrutura em árvore ou "hash", tendo como chaves os próprios identificadores. Para cada classe é mantida uma árvore independente das demais. Dessa forma não só é possível manter os identificadores válidos apenas no contexto de cada classe, como é possível manter juntas as informações referentes aos identificadores de cada classe acessada. Tal como no caso da Base de Dados do SACDS, cada página deve conter apenas uma sub-árvore.

-- Arquivo de Objetos: armazena as informações referentes a cada objeto da descrição. Devem ser mantidos em cada página os objetos cujos identificadores pertencem a uma mesma classe.

-- Arquivo de Relacionamentos: armazena as informações sobre todos os relacionamentos entre objetos existentes na descrição. Da mesma maneira que na Base de Dados do SACDS, deve ser estruturado em blocos, sendo que cada bloco contém relacionamentos referentes a um mesmo objeto, e da mesma maneira que naquele caso, o tamanho de cada bloco deve ser atribuído de maneira que se obtenha a melhor relação registros-não-usados/dispersão-das-informações.

-- Arquivo de Atributos: armazena os ponteiros de acesso aos atributos de cada objeto ou relacionamento, de cada característica de atributos, tanto dos atributos de cada objeto ou relacionamento em si, como de todos os atributos "herdados".

Cada página desse arquivo deve armazenar os atributos: dos objetos cujos identificadores pertencem a uma mesma classe; dos

relacionamentos que existem entre dois objetos cujos identificadores pertencem a uma mesma classe; dos relacionamentos que existem entre os objetos cujos identificadores pertencem a uma mesma classe e objetos cujos identificadores não pertencem a essa classe. Os atributos que são armazenados em cada página não devem pertencer a mais do que um dos itens acima.

-- Arquivos de Atributos com características especiais (Propriedades, Comentários, Regras, etc.): armazenam os atributos de cada característica, de cada objeto ou relacionamento. Os arquivos de atributos devem ser estruturados da mesma maneira que o arquivo de relacionamentos, com cada bloco armazenando os atributos da característica correspondente ao arquivo, de um determinado objeto ou relacionamento.

Para tratar de todos esses tipos de entidades, os Processadores de Manipulação devem conter programas especiais para cada um dos tipos de entidades, e cada uma das operações que são necessárias. De uma maneira geral, todos os arquivos demandam os mesmos recursos computacionais, e a mesma quantidade de memória, a menos do arquivo de Identificadores.

As operações necessárias sobre esse arquivo incluem as que usualmente ocorrem nos demais arquivos (por exemplo, consultar qual é a cadeia de caracteres que constitui o identificador corrente), porém existem além dessas, as operações que não são acompanhadas de um código, mas sim que solicitam o reconhecimento (ou a colocação) de um identificador, através de uma cadeia de caracteres. Essas operações fornecem, no pacote que transmite o pedido, apenas a cadeia de caracteres procurada e o código da

classe onde a busca será feita. O código passado nesse caso não diz respeito diretamente à página onde a busca será feita, mas apenas ao registro onde o ponteiro inicial das páginas que potencialmente podem armazenar a informação procurada. O processador que atende a esse tipo de solicitação necessita assim de acesso a muito mais memória, ao mesmo tempo em que os índices das páginas a que ele tem acesso num dado instante não são importantes para determinar a aceitação de um pedido. Dessa forma, é possível admitir a especialização de Processadores de Manipulação, para melhorar o desempenho geral da máquina, atribuindo funções e recursos (memória) adequados a cada Processador de Manipulação.

Assim, embora em uma implementação inicial possam existir apenas Processadores de Manipulação idênticos, é possível admitir que uma implementação mais refinada apresente Processadores de Manipulação especiais para cada uma das categorias abaixo:

- Processadores de Manipulação de Identificadores: dispõem de mais bancos de memória, com um tamanho de página adequado para esse arquivo (provavelmente maiores que 8 KBytes);
- Processadores de Manipulação de Relacionamentos e Atributos: dispõem de um número menor de bancos, e tamanho de página também adequado a esses arquivos (provavelmente menores que 8 KBytes);
- Processadores de Manipulação de Objetos, Classes e Atributos com características especiais: os mesmos Processadores que seriam usados em uma máquina com processadores não especializados;
- Processadores de Manipulação de Seleções: não necessitam de comparadores de códigos em suas respectivas interfaces, e atuariam no sentido de obter dos demais processadores,

através de um conjunto de vários pedidos de acesso, todas as informações que constituem e envolvem seleções de objetos (em princípio, todas as operações apresentadas na seção IV.4.1.2.).

A diferença entre os vários tipos de Processadores de Manipulação seria a princípio apenas na capacidade de memória de cada um, e como isso acarreta uma modificação no número de páginas que podem ser armazenadas, haveria uma variação no número de registradores que armazenam o número de índices de páginas que deveriam ser comparadas na Interface com o Duto de Dados.

Como um Processador de Manipulação de Identificadores necessita mais memória, mas correspondentemente o tamanho de cada página também seria melhor se fosse maior, deve se verificar se o número de páginas não poderia ser mantido constante. Nesse caso o número de registradores para índices de páginas na Interface com o Duto de Dados poderia ser sempre o mesmo. O mesmo ocorre com os Processadores de Manipulação de Relacionamentos e Atributos, apenas com páginas menores, e correspondentemente menos memória.

Se essa última forma se mostrar conveniente, o Controlador de Discos deverá levar em conta o tamanho de página para as transferências para cada tipo de Processador de Manipulação, porém o número de bits comparados nas Interfaces com o Duto de Dados de qualquer tipo de Processador de Manipulação será sempre constante.

Uma vantagem da existência de Processadores de Manipulação especializados é a maior facilidade de se gerenciar os programas que devem ser executados nos vários processadores. Os programas a serem executados na Máquina proposta seriam a princípio carrega-

dos através do Duto de Páginas. Processadores não diferenciados deverão ser capazes de executar qualquer uma das operações que um Processador de Manipulação deva executar. Como não se espera que todos os Processadores tenham memória para armazenar o código de todas as operações possíveis, sempre que um processador necessitar executar uma operação para a qual não tenha o código disponível, este deverá então ser carregado. Isso ocasiona a necessidade de se gerenciar também a memória de programas de cada Processador de Manipulação, além de contribuir para aumentar a carga do Duto de Dados. Havendo Processadores de Manipulação especializados, pode-se prever que muitos programas ficarão permanentemente na memória dos Processadores que os executam. Esses programas poderiam então ser carregados apenas no início de operação da Máquina, através de um procedimento simples, que dispensa o gerenciamento dinâmico, e quando a máquina já estiver em execução normal, não será mais usado o Duto de Páginas para a transferência de programas.

Os Processadores de Manipulação de Seleções deveriam dispôr de uma memória para programas maior, e talvez pudessem dispôr de uma unidade de disco específica para o carregamento desses programas, uma vez que poderiam executar uma variedade de operações muito maior.

Todas as operações de manipulação propostas no capítulo IV deste trabalho, que não as que envolvem Seleções (que são as que se sugere que sejam atendidas pelos Processadores de Manipulação, e não os de Seleções), são operações que atendem o acesso a apenas uma entidade de cada vez. Os Processadores de Manipulação de Seleções atenderiam aos pedidos que envolvem conjuntos de entidades de mesmo tipo, que correspondem a Seleções de Objetos.

Se bem que as mesmas operações seriam realizadas por processadores não especializados, a existência de Processadores de Manipulação específicos para o tratamento de Seleções de Objetos permite que essas operações não entrem em concorrência com as demais, permitindo que esses processadores possam aproveitar tempos ociosos para a busca prévia de operações com alta probabilidade de serem solicitadas posteriormente. Além disso, esses processadores seriam úteis para fazer a Base de Dados responder como se estivesse apoiada em outros modelos de dados, tais como o modelo relacional, hierárquico ou de rede.

VII.8. - Conclusões.

Neste trabalho foram discutidas duas máquinas voltadas para o processamento das tarefas de Gerenciamento de uma Base de Dados. A primeira (Máquina SACDS/SGBD) apresentada no capítulo VI, foi efetivamente implementada em termos de seus componentes de "Hardware", ao passo que a segunda máquina, apresentada neste capítulo, constitui-se em uma proposta de uma Máquina de Base de Dados, cuja concepção foi motivada pelo trabalho de desenvolvimento da anterior.

A Máquina SACDS/SGBD apresenta uma arquitetura que permite a implementação otimizada do SACDS, aproveitando suas características de processamento paralelo. Além disso, pode ser utilizada como um Sistema de Multi-micro-processadores, executando programas normalmente desenvolvidos para um sistema operacional de propósito geral, quando então pode dispor de até quatro estações

de trabalho acessando o processador central como um gerenciador de arquivos.

A concepção inicial que motivou o desenvolvimento dessa máquina, era a de que apenas um programa de aplicação em execução seria o SACDS, e portanto haveria apenas uma estação de trabalho acessando a base de dados ao mesmo tempo. Do desenvolvimento do Modelo de Representação de Objetos, e da constatação que muitas Estações de Trabalho deveriam poder acessar a base de dados simultaneamente, surgiram as considerações para a proposta da nova máquina.

O sistema de software completo, necessário para que a máquina já disponível pudesse ser efetivamente usada para prestar serviços de Gerenciamento de Bases de Dados, não chegou a ser concluído, principalmente devido ao fato de que a muito maior utilidade de um sistema que permitisse o acesso de vários programas de aplicação em execução simultânea somente poderia ser eficientemente implementado em uma máquina com mais recursos do que a que fora originalmente projetada e foi construída.

Assim o conjunto de instruções que permitiriam a execução do SGBD do SACDS no Processador Central da máquina proposta não foi completado, porque da experiência obtida durante o desenvolvimento da máquina descrita, e do desenvolvimento de trabalhos relativos ao próprio SACDS, foi concebido o Modelo de Representação de Objetos (MRO), o qual passou a representar um fator adicional de desenvolvimento do próprio SACDS. Além disso, da experiência com o desenvolvimento da máquina descrita no capítulo VI, e da concepção de novos recursos que poderiam ser-lhe adicionados, sugeridos pela avaliação do seu desempenho, a máquina sugerida neste capítulo foi concebida. A partir daí, um novo objetivo

delineou-se, tornando-se claro que o conjunto de instruções a ser desenvolvido deveria ser o que permitisse a execução de um Sistema de Gerenciamento de Bases de Dados apoiado no Modelo de Representação de Objetos.

Este trabalho relata o estado de desenvolvimento a que se chegou até aqui, e os protótipos, tanto de hardware quanto de software, que permitiram que as idéias apresentadas tomassem forma. No entanto, agora uma nova fase está bem clara, e este trabalho pode ser considerado o encerramento da fase anterior. A continuação desse trabalho, deve empregar os recursos disponíveis na implementação da nova máquina sugerida e na implementação de um SGBD apoiado no MRO, e não mais no refinamento dos protótipos que serviram para a sua concepção.

C A P Í T U L O V I I I

CONCLUSÕES E LINHAS DE FUTURAS PESQUISAS

VIII.1. - Considerações Gerais

Neste capítulo são apresentadas idéias e sugestões que podem dar prosseguimento ao trabalho ora apresentado. Pode-se dizer que este trabalho consiste essencialmente na apresentação de um modelo de dados, ao qual se anuiu a existência de características para apoiar o desenvolvimento de Sistemas de Gerenciamento de Dados para Engenharia, e adequado para execução em sistemas de Multiprocessamento.

Uma vez completada essa apresentação, a sua adequação para satisfazer as necessidades de um Sistema de Gerenciamento de Dados para Engenharia foi verificada ao final do capítulo IV, quando se fez a confrontação com as necessidades de um tal sistema, as quais foram levantadas no capítulo II. Resta agora considerar como esse modelo se apresenta frente às necessidades de outras aplicações, notadamente àquelas normalmente atendidas por modelos de dados ditos convencionais.

Além disso, o modelo proposto é bastante maleável quanto à possibilidade de incorporar novos recursos que se mostrem neces-

sários, bem como pode ser restringido para apresentar uma complexidade reduzida em aplicações que não demandem características não essenciais. Isso abre campo tanto para o estudo de novos recursos e técnicas que lhe possam ser incorporadas, quanto para o estudo de uma forma de sistematizar a análise de recursos que devem ser incluídos em Sistemas de Gerenciamento de Bases de Dados que usem esse modelo, e por extensão, para o estudo de formas que permitam sistematizar a identificação de recursos que devem ser supridos por SGBDs em geral para atender as necessidades de aplicações específicas.

Por outro lado, algumas soluções apresentadas estão condizentes com determinados compromissos entre eficiência, capacidade de representação, complexidade e formas de acesso esperadas. Muitos desses compromissos podem ser modificados, quando então as soluções apresentadas devem ser reavaliadas.

Nas seções seguintes são discutidos esses três pontos, ao mesmo tempo em que sugerem-se pesquisas que podem vir a esclarecer ou desenvolver tópicos de interesse para este trabalho, e que podem dar continuidade ao desenvolvimento do modelo apresentado.

VIII.2. - O Uso de uma Máquina de Base de Dados Específica.

A máquina de base de dados proposta no capítulo V, por si só já se constitui em uma proposta de desenvolvimento, qual seja, a sua implementação efetiva. Vários tópicos interessantes decorrem daí, tal como sugere-se naquele capítulo, porém consideramos que devido ao grande esforço necessário para a realização dessa

máquina, um modelo dela deve ser desenvolvido, de maneira a permitir simulações de várias configurações possíveis, sob diversas modalidades de carga de trabalho.

Alterações possíveis nessa máquina poderiam então ser simuladas, para que as melhores alternativas de implementação sejam determinadas com base em dados estatísticos. Uma possível alteração que deveria ser simulada corresponde a fazer com que toda solicitação de pedidos de acesso para a base de dados seja tratada diretamente pelo processador de manipulação, o qual faria então com que todos os pacotes de dados circulando pelo duto de dados tivessem sempre um processador destino determinado. Outros pontos que seriam obtidos da simulação, seria a localização de pontos de "Gargalo" que deveriam então ser estudados para que se minimizasse o efeito na degradação da performance da máquina.

Outro ponto que deveria ser levado em conta corresponde a estabelecer quando as restrições devidas às operações de alto nível semântico deveriam ser feitas. Esse ponto foi discutido na seção III.7, onde se chegou a conclusão que a verificação de regras de consistência e completeza seria executada de preferência ao final da execução dos programas que efetuassem alterações na base de dados. Porém, a existência de múltiplos processadores que podem manipular os dados solicitados pelo programa, levanta a questão de se não seria preferível antecipar essa verificação para o instante em que os dados presentes na memória de um processador devem ser substituídos por outro. O fato de se executar essa última operação, significa que os dados já não estão sendo solicitados pelo programa, o que por sua vez indica que a verificação já poderia ser empreendida, e a sua retirada da memória, com operações de verificação pendentes, indica que os

dados deverão ser posteriormente recarregados.

VIII.3. - O Uso do Modelo de Representação de Objetos em Aplicações que Não de Projeto.

Uma característica que permeia todo o Modelo de Representação de Objetos é a sua polarização para aplicações que se caracterizam como de apoio a projetos e a informações com baixa ocorrência de estruturas repetitivas. A baixa repetição de estruturas idênticas faz com que se ressalte a necessidade de identificação dos dados armazenados. Dessa forma, um dos pontos em que o modelo apoia-se é na existência de um identificador individual e único, reconhecido em geral pelo próprio usuário dos programas de aplicação que se utilizam do modelo, que pode permitir o reconhecimento e o acesso aos dados armazenados.

Devido a isso, no MRD toda a informação armazenada é acessada diretamente através de um identificador, e a seguir, efetuam-se operações de navegação na base de dados. Essa característica é típica de sistemas de apoio a projeto, onde todos os detalhes são individualmente tratados, e recebem por isso identificação própria. As propriedades e dados de cada item são associados a cada um de forma a permitir o rastreamento das informações.

Em aplicações que mantêm uma alta regularidade nas estruturas de dados, tais como aquelas que são normalmente atendidas pelos modelos de dados tradicionais, o MRD já não se apresenta adequado. Isso deve-se principalmente à inexistência lógica de identificadores individuais para cada um dos dados armazenados, o

que torna difícil a recuperação dos dados, e pela diferente forma predominante de acesso a esses dados. Vejamos o porquê de cada uma dessas duas afirmações, comparando o MRO com o Modelo Relacional em uma aplicação típica para este último.

Considere-se um sistema de controle de estoque, onde as ordens de compra devem ser mantidas na Base de Dados, de uma maneira semelhante ao exemplo usado para ilustrar a apresentação do Modelo Relacional no capítulo 2, e que aparece na Figura 5. Nessa figura, pode-se perceber que a recuperação de um registro nem sempre pode ser feito por um nome que se associa a ele. Por exemplo, apesar de todas as ordens de compra possuírem um número, e todos os produtos poderem ser designados por nomes, na Relação Produtos daquela figura cada registro somente é identificado univocamente pela indicação do número da Ordem de Compra e pelo número do produto, não existindo nenhum identificador individual para qualquer registro.

Além disso, o acesso às informações nessa aplicação, nem sempre é feito através de dados aos quais se possa, mesmo de maneira composta, estabelecer um identificador (tal como número do produto numa ordem de compra). É comum por exemplo que se solicitem todos os produtos com quantidade pedida acima de determinado valor.

Se o MRO for usado para armazenar os dados dessa aplicação, a grosso modo pode-se dizer que existiriam objetos do tipo Ordem-de-Compra, Fornecedor e Produtos. Nesse caso, os atributos da Relação-de-Produtos-por-Ordem-de-Compra da Figura 5 seriam propriedades de um Relacionamento tipo Compra entre objetos do tipo Ordem-de-Compra e Produtos. O MRO apresentaria vantagem para uma consulta tal como obter todos os produtos pedidos em uma determi-

nada ordem de compra (bastaria obter todos os objetos destino dos relacionamentos Compra que tem a Ordem de Compra indicada como objeto origem), porém estaria em desvantagem para uma consulta tal como obter todos os produtos com quantidade pedida acima de determinado valor (em que seria necessário verificar a propriedade Quantidade-Pedida em todos os relacionamentos do tipo Compra em todas as Ordens de Compra). Nesse caso, a estrutura mais simples do Modelo Relacional apresentaria vantagem para a pesquisa, em relação à estrutura mais elaborada do MRD.

Como nas aplicações usuais do Modelo Relacional, as consultas dos dois tipos apresentados acima podem ocorrer indistintamente, devem ser efetuadas avaliações mais apuradas de aplicações do MRD em aplicações que não de Projeto, principalmente em comparação com outros modelos.

Um outro ponto que deve ser estudado quanto à utilização do MRD em outras aplicações consiste no próprio formato dos atributos de identificação. No capítulo III foi considerado que para uma aplicação que requisesse um Meta-Sistema de Gerenciamento de Dados, seria necessário que todos os atributos de identificação tivessem o mesmo formato, daí a existência de apenas um arquivo de identificadores de objetos. Porém, se for considerado que cada tipo de objeto possa ter um identificador diferente, então poderão existir muitos arquivos de identificadores de objetos, cada um adequado para o formato de identificador que armazenar. Por exemplo, uma aplicação de Recuperação de Informação Bibliográfica, onde existiriam como tipos de objetos Autor, Livro, Artigo, Editora, etc, pode-se perceber que a identificação de um Autor necessita de muito mais caracteres do que o Código de um Livro,

ou muito menos do que o Título de um Artigo.

A dificuldade inerente a um enfoque desse tipo (e que deve ser melhor analisada), consiste na aplicação de atributos com característica de Sinônimos a objetos com variados formatos para identificadores. No mesmo exemplo acima, se um Livro pudesse ter tanto Código quanto Título, e esses dois identificadores tivessem formatos diferentes, a especificação de um identificador para objetos de determinado tipo não é suficiente, pois torna-se necessário a indicação do tipo de sinônimo com que cada identificação é feita. Isso corresponde a alterar todas as rotinas que envolvem objetos e/ou sinônimos apresentadas no capítulo IV, bem como devem ser avaliadas todas as possíveis consequências que afetem a semântica do MRD.

VIII.4. - Desenvolvimento de Ferramental Matemático para Análise do Modelo de Representação de Objetos.

O Modelo de Dados para implementação de Bases de Dados que é mais estudado e aplicado atualmente (em termos de diversidade de aplicações) é sem dúvida o Modelo Relacional. Uma de suas grandes forças está na disponibilidade de um Modelo Matemático que permite analisar qualquer aplicação em termos de suas componentes básicas de dados segundo critérios bem conhecidos e seguros. Assim, existem procedimentos bem determinados para efetuar-se a modelagem de uma aplicação qualquer para os requisitos do Modelo Relacional, em termos de suas operações, dependências de dados, casos omissos, etc. Além disso, os efeitos de alterações na Base

de Dados decorrentes de alterações solicitadas segundo diversas visões de dados podem ser previstas e analisadas.

O Modelo Entidade-Relacionamento pode ser visto, em termos de identificação de um modelo matemático para o Modelo de Dados, como uma extensão ao Modelo Relacional, onde as relações do primeiro são conceitualmente divididas em relações de entidades e relações de relacionamentos, sendo que relações de entidades e relações de relacionamentos, todas elas mantêm as mesmas propriedades de relações tal como qualquer relação no Modelo Relacional, porém incorporam mais propriedades próprias conforme sejam relações de entidades ou relações de relacionamentos. Essa alteração no modelo, com a correspondente incorporação de propriedades, devem necessariamente ser representadas em um modelo matemático. Isso tem sido desenvolvido e apresentado na literatura sob variados aspectos [NG*_81] [JAJ_83] [PAR_85], porém a complexidade adicional do modelo torna muito mais difícil a identificação inequívoca de aspectos de uma aplicação com conceitos de qualquer modelo matemático já desenvolvido.

Seguindo a mesma linha de raciocínio, deve-se esperar que o desenvolvimento de um modelo matemático para o MRD traria inúmeros benefícios para a aplicação do modelo, porém deve-se esperar também que a complexidade ainda maior do MRD dificulte ainda mais a identificação de situações de representação de dados de cada aplicação com as estruturas do MRD. Assim, ao menos para um primeiro enfoque, seria de interesse o desenvolvimento de um modelo matemático que permitisse representar seletivamente as principais características de uma aplicação para a qual se intentasse o uso do MRD.

Para o desenvolvimento desse modelo deve-se levar em conta

os principais conceitos que constituem o MRD:

- Toda a informação do mundo real é representada através da identificação dos objetos do mundo real para os quais exista interesse em se representar;
- Todos os relacionamentos que existem entre objetos devem ser representados no modelo através de relacionamentos entre esses objetos;
- Relacionamentos que envolvem mais de dois objetos são compostos de relacionamentos que envolvem os objetos dois a dois;
- Tanto os objetos quanto os relacionamentos podem ter atributos;
- Os atributos possuem características semânticas que os individualizam, bem como as operações que podem ser solicitadas individualmente sobre eles;
- Os objetos devem necessariamente ser identificados por atributos com característica de sinônimos ou através de relacionamentos;
- Os atributos de identificação de objetos são seletivos na abrangência de identificação, fazendo com que o nível de especificidade de um identificador varie conforme o grau de detalhe com que se indicam objetos para consideração;
- Atributos com característica de sinônimo não podem ser associados a relacionamentos, mas a menos desses, quaisquer outros sinônimos podem ser indistintamente associados a objetos ou a relacionamentos;
- Todos os objetos, relacionamentos e atributos são caracterizados através de seus tipos;

- Os objetos podem ter subtipos, que são caracterizados por valores de determinados atributos que lhes estão associados;
- Conjuntos de tipos de objetos podem caracterizar supertipos de objetos;
- Os relacionamentos que existem entre objetos de determinados tipos e seus supertipos, bem como atributos desses objetos e do relacionamento podem ser herdados pelos objetos que pertencem ao conjunto de objetos que compõem um supertipo.

Para o desenvolvimento de um tal modelo matemático, não deve levar-se em consideração quais são as características de atributos suportadas por um modelo, mas apenas o fato de que várias podem ser suportadas indistintamente, e que atributos com característica de sinônimos sempre existirão, com a propriedade adicional de somente serem associados a objetos.

Dois devem ser os objetivos principais desse modelo: identificação de operações atendidas pelo SGBD; e análise do modelo conceitual de uma aplicação, para a identificação da correspondência entre as operações solicitadas sobre cada visão de dados com as operações realmente efetuadas na base de dados.

A identificação de operações que devem ser atendidas por um SGBD, apoiado no MRO é importante para se determinar um conjunto mínimo de operações que devem ser supridas, bem como para se determinar que não existam efeitos deletérios entre operações em execução paralela, de maneira a se poder estabelecer critérios para permitir a execução paralela de vários programas de aplicação simultaneamente sobre uma mesma base de dados. O conjunto de

operações proposto no capítulo IV deveria ser então reavaliado e a ele incorporadas as considerações sobre o paralelismo possível entre operações.

A análise do modelo conceitual de uma aplicação, com vistas a determinar que operações devem efetivamente ser levadas a efeito na base de dados, a partir de operações solicitadas segundo uma particular visão de dados de um programa de aplicação, constitui o principal objetivo de um tal modelo, uma vez que a diversidade de opções de representação de dados causa ambigüidade na definição e uso dos recursos do modelo. Assim, além da possível equivalência entre atributos de relacionamentos com valores discretos e vários tipos de relacionamentos (o que já ocorre no ME-R), no MRO existem ambigüidades na escolha de quais tipos de objetos irão constituir determinadas classes de objetos, associação de atributos a tipos de objetos ou a subtipos, relacionamentos entre objetos ou entre subtipos, identificação de objetos através de relacionamentos ou derivação de nomes de objetos a partir de relacionamentos, etc. A elaboração de critérios que permitam mapear cada construção em outras construções equivalentes é fundamental para permitir o uso do modelo em situações em que ocorram conflitos nas várias visões de usuários que podem constituir um esquema conceitual.

Um outro ponto que pode ser considerado interessante do ponto de vista de uma possível modelagem, corresponde à determinação de operações que podem ser executadas sobre os dados armazenados, e a sua respectiva indicação na Meta-base de Dados. Esse ponto, que poderia ser visto como um conjunto de informações sobre a Dinâmica dos dados (uma vez que diz respeito à forma como os dados são modificados), tem ainda mais importância no MRO do

que em outros modelos, uma vez que constitui-se em informação que aumenta o significado dos dados para a Base de Dados. Sendo um modelo de alto conteúdo semântico, a disponibilidade de informações que permitam ao SGBD gerenciar as operações que podem ser efetuadas sobre dados armazenados de uma forma mais refinada, constitui um elemento importante para o MRD.

A possibilidade de se incorporar essas informações na Meta Base de Dados teria por objetivo verificar como cada uma das operações básicas (inclusão, alteração, ou apagamento) poderia ser realizada sobre cada um dos elementos de dados que constitui uma base de dados (cada tipo de objeto, cada tipo de atributo de cada tipo de objeto, cada relacionamento, com cada tipo de objeto destino, etc.). Assim, por exemplo, poder-se-ia admitir que o próprio SGBD impedisse alterações em características de um objeto do tipo TIPO-DE-CI depois que já existissem na base de dados objetos tipo PLACA-DE-CI que usassem aquele objeto.

VIII.5. - Alterações na Estrutura de Dados Independente do Caminho de Acesso no Modelo de Representação de Objetos.

A discussão sobre o MRD contida no capítulo IV deste trabalho, traz uma série de considerações sobre o modelo baseadas na forma como um protótipo de um SGBD nele apoiado foi implementado. Esse protótipo foi desenvolvido tendo como objetivo suprir necessidades específicas, que ditaram em primeiro lugar a estrutura de dados independente do caminho de acesso adotadas, e

conduziram mesmo, em alguns casos, ao desenvolvimento dos próprios conceitos apresentados. Por outro lado, muitos dos compromissos assumidos não são universalmente válidos, e um estudo sobre diferentes alternativas de implementação devem ser considerados. Um exemplo de situações em que isso ocorre foi discutido na seção VIII.3, onde sugeriu-se que a existência de apenas um arquivo de identificadores de objetos poderia ser re-estudada.

Outros aspectos bastante ligados à própria estrutura de dados dependente do caminho de acesso poderiam também ser objeto de um estudo mais profundo. Um exemplo disso é a sugestão de se usar estruturas de árvore binária para a recuperação de identificadores no arquivo de identificadores. Essa sugestão decorre de estudos efetuados para a implementação do protótipo citado, e de discussões ocorridas no desenvolvimento do Sistema de Gerenciamento de Base de Dados para o SIPS (mencionado no capítulo I) do Centro Tecnológico para Informática. Nessa discussão, optou-se pelo uso dessa mesma estrutura, embora tenha sido considerado que técnicas como partição ("Hash") composta seriam igualmente apropriadas para aquele caso. O uso de árvores no entanto parece ser preferível quando leva-se em conta a possibilidade de execução do SGBD em Máquinas de Base de Dados como aquela proposta no capítulo 5.

Um aspecto que deve ser considerado corresponde às estruturas de representação de atributos que são "herdados" devido aos conceitos de tipo e subtipo de objeto, e aos que são herdados devido a para-relacionamentos existentes entre objetos e seus supertipos. Segundo o modelo apresentado no capítulo IV, esses dados são armazenados através de associações implícitas entre os objetos de interesse e o objeto a que o conceito de herança

permite legar atributos e relacionamentos. Em aplicações em que a quantidade de associações desse tipo se mostrarem elevadas, é possível impor uma estrutura que explicita essas associações. A existência de um arquivo que mantém os cabeçalhos de todos os atributos na estrutura do MRO apresentado pode servir como um ponto de partida para isso; porém devem ser analisadas as formas que permitam armazenar as associações com todas as implicações que esses conceitos apresentam.

Um exemplo de consequência que deve ser levado em conta diz respeito a forma como é feita a passagem de atributos e relacionamentos herdados entre diversas versões. Um exemplo que permite entender o significado desse ponto corresponde à documentação associada a cada versão. Esta é mantida na Base de Dados através dos atributos com característica de Comentário que são associados a objetos específicos. Através de mecanismos de herança de comentários, a documentação de uma determinada versão pode ser "montada" a partir dos atributos espalhados em vários níveis de versões e alternativas, permitindo que a documentação tanto acompanhe os dados armazenados, quanto crie meios para que a alteração de dados seja acompanhada automaticamente de alterações na documentação. Recuperação de históricos de desenvolvimento por exemplo, são atividades que se beneficiam dessa característica, e podem ser tomados como base de teste para a avaliação de mecanismos de herança explícita de atributos.

VIII.6. - Considerações Sobre o Significado do que é Armazenado pelo Modelo de Representação de Objetos

A disponibilidade de recursos de representação de dados oferecida pelo MRD é bastante grande, como não poderia deixar de ser, dado que uma gama bastante vasta de situações do mundo real devem ser atendidas por ele. Porém, isso pode levar tanto a situações de ambigüidades nas possibilidades de esquematização das estruturas de dados, quanto nas operações que se oferecem aos programas de aplicação. Muitos desses casos poderiam ser analisados e dispor de uma solução padronizada se um Modelo Matemático tal como o sugerido na seção VIII.4 pudesse se tornar disponível. Porém, algumas operações sugeridas no capítulo IV não seriam cobertas por um modelo, por mais abrangente que este fosse. Tal é o caso das operações de inclusão automática de relacionamentos e atributos.

Numa operação desse tipo, o programa de aplicação deve passar ao SGBD instruções sobre quais relacionamentos e atributos colocar, e sob que condições isso deve ser efetuado. Operando sob solicitações explícitas de um programa de aplicação, as operações de inclusão automática de dados ficam associadas ao particular programa de aplicação. Quando efetivamente o programa é repositório dessas informações, então pode-se considerar que a informação está distribuída entre o programa e a base de dados, e as operações que são solicitadas para execução automática são semanticamente corretas. Porém, quando isso não ocorre, então as informações usadas para indicar as operações automáticas deveriam estar armazenadas na própria base de dados.

As informações podem ser colocadas na base de dados de

várias maneiras, porém pode-se prever que em geral deverá existir um objeto que seja responsável por elas e as mantenha. Esse é o caso de um objeto que tenha um supertipo, e portanto resuma as informações de um conjunto de objetos. Assim, se ao invés de uma aplicação passar ao SGBD um conjunto de solicitações de inclusão de relacionamentos e atributos automaticamente, ela passar a definição de um objeto com um determinado supertipo, ao qual estão associados aqueles atributos e relacionamentos, então um efeito semelhante irá ocorrer, porém as informações ficam permanentemente armazenadas na base de dados, ao passo que no primeiro caso elas apenas estariam disponíveis ao SGBD durante aquela específica transação.

Verifica-se assim que existe necessidade de se analisar não somente o significado dos dados que são armazenados na base de dados, mas também as operações que podem ser solicitadas ao SGBD. Se bem que a ampla gama de opções e casos em que isso possa ocorrer tornem impossível a análise prévia exaustiva, ainda assim o levantamento de um conjunto dos casos mais comuns poderia ser encetado, e sua análise poderia servir tanto para a especificação dos casos analisados, quanto para orientação de como agir nos casos não analisados.

VIII.7. - Continuação do Aumento de Capacidade de Representação de Semântica do Modelo.

As informações sintáticas de um sistema sempre podem ser totalmente descritas e armazenadas em um modelo de representação,

o mesmo não ocorre com as informações semânticas. Assim, as informações descritas no capítulo IV como o conjunto de informações semânticas do MRO não representam um conjunto fechado, que não possa ser modificado.

Na realidade, o modelo apresentado neste trabalho constitui-se em um modelo acabado, no sentido de que é completo para representar determinadas situações do mundo real, possuindo um conjunto de operações que permitem manipular as informações que representa de maneira consistente. Porém, se for considerado que é desejável poder aumentar a capacidade de representação semântica do modelo, então pode-se considerar que o modelo está aberto a expansões que venham a ser úteis para a representação de situações que não estejam ainda cobertas.

Uma possibilidade interessante para a detecção de situações assim, e que seria também de muita valia para o próprio desenvolvimento e manutenção de implementações de SGBDs usando o MRO, seria a representação do modelo usando a sua própria estrutura. Isso pode ser feito se for considerado que o modelo em si é um sistema, e como tal pode ser modelado e representado. Especialmente interessante seria a modelagem de toda a semântica básica, pois não somente forneceria uma indicação precisa de como as implementações dos SGBDs deveriam ser feitas, como permitiria uma análise estruturada de consistência e completeza da semântica representada.

Alguns casos de representação de informação reconhecidos na literatura levantada no capítulo II e que não foram tratados neste trabalho deveriam também ser analisados. Desses, o caso mais expressivo talvez seja a possibilidade de se armazenar junto ao valor de um atributo com característica de propriedade a

informação de precisão dessa informação.

Generalizando esse conceito, pode-se pensar que os valores das propriedades são armazenados com o significado implícito que a propriedade é igual ao valor armazenado. Poder-se-ia admitir que um valor fosse armazenado com a informação que a propriedade é menor, ou maior ou menor ou igual, etc. que o valor armazenado, ou mesmo que este está entre uma faixa de valores. Dessa forma, a dificuldade em se estabelecer um valor para o atributo que define o subtipo de um objeto sendo definido, quando o predicado de definição consiste de uma expressão de desigualdade, teria sido eliminada, uma vez que a própria expressão teria toda a informação sobre qual o valor, e qual a condição associada ao valor deveriam ser armazenados.

B I B L I O G R A F I A

- [AGR_85] Agrawal, R. - "A Paralell Logging Algoritm for Multiprocessor Database Machines", in Proc. 4th Int. Workshop on Database Machines, March 1985, pp. 256-276.
- [ALM_85] Almes, G. T. et al. - "The Eden System: A Tecnical Review", IEEE Trans. Software Engineering, Vol. SE-11 Nro. 1, January 85, pp. 43-59.
- [AST_76] Astrahan, M. M. - "System R: Relational Approach to Data Base Management", ACM Trans. Database Systems, Vol. 1, Nro. 2, June 76, pp. 97-137.
- [BAB_79] Babb, E. - "Implementing a Relational Database by Means of Specialized Hardware", ACM TODS, Vol. 4, March 1979, pp. 1-29.
- [BAC_69] Bachman, C. W. - "Data Structure Diagrams", Data Base, J. ACM SIGBDP, Vol. 1, Nro. 2, pp. 4-10.
- [BAN_78] Banerjee, J.; Baum, R. I.; Hsiao, D. K.; Kannan, K. - "Concepts and Capabilities of a Database Computer", ACM TODS, Vol. 3. Nro. 4, December 79, pp. 347-384.
- [BAN_79] Banerjee, J.; Hsiao, D. K.; Kannan, K. - "DBC - A Database Computer for Very Large Databases", IEEE Trans. on Computers, Vol. C-28, Nro. 6, June 1979, pp. 414-429.
- [BEC_80] Becerril, J. L. et al. - "GSYSR: A Relational Database Interfase for Graphics", Lecture Notes in Computing Science, Vol. 81, New York: Springer-Verlag, 1980, pp. 459-474.
- [BER_84] Bersoff, E. - "Elements of Software Configuration Management", IEEE Trans. Software Engineering, Vol. SE-10, Nro. 1, January 84, pp. 79-87.
- [BIC_85] Bic, L.; Hartmann, R. L. - "Hither Hundreds of Processors in a Database Machine", in Proc. 4th Int. Workshop on Database Machines, March 1985, pp. 153-168.
- [BIC_86] Bic, L.; Gilbert, J. P. - "Learning from AI: New Trends in Database Technology", IEEE Computer, Vol. 19, Nro. 1, January 86, pp. 44-54.
- [BIT_83] Bitton, D.; Boral, H.; DeWitt, D. J.; Wilkinson, W. K. - "Paralell Algoritms for the Execution of Relational Database Operations", ACM Trans. on Database Systems, Vol. 8, Nro. 3, September 83, pp. 324-353.

- [BOR_81] Boral, H.; DeWitt, D. J. - "Processor Allocation Strategies for Multiprocessor Database Machines", ACM TODS, Vol. 6., June 81.
- [BRA_81] Bradley, J - "File & Data Base Techniques" CBS College Publishing - HRW, 1981.
- [BRH_83] Brachman, W. A. - "What IS-A and Isn't: An analysis of Taxonomic Links in Semantic Networks", IEEE Computer, Vol. 16, Nro. 10, October 83, pp. 30-36.
- [BRO_85] Browne, J. C.; Dale, A. G.; Leung, C.; Jenevein, R. - "A Parallel Multi-stage I/O Architecture with Self-Managing Disk Cache for Database Management Applications", in Proc. 4th Int. Workshop on Database Machines, March 1985, pp. 331-346.
- [CAN_74] Canaday, R. H. et alli - "A Backend Computer for Database Management", Commun. ACM, Vol. 17, Nro. 10, October 74, pp. 575-582.
- [CES_85] Cesarini, F.; Pippolini, F.; Soda, G. - "A Technique for Analyzing Query Execution in a Multiprocessor Database Machine", in Proc. 4th Int. Workshop on Database Machines, March 1985, pp. 68-90.
- [CHA_80] Chang, - "A Relational Database System for Images", in Lecture Notes in Computer Science, Vol. 80, New York: Springer-Verlag, 1980.
- [CHE_76] Chen, P. P. - "The Entity-Relationship Model - Toward a Unified View of Data" ACM Trans. Database Systems, Vol. 1, Nro. 1, march 76, pp. 9-36.
- [COD_70] Codd, E. F. - "A Relational Model for Large Shared Data Banks", Commun. ACM, Vol. 13, Nro. 6, June 70, pp. 377-387.
- [COD_79] Codd, E. F. - "Extending the Database Relational Model to Capture More Meaning", ACM Trans. Database Systems, Vol. 4, Nro. 4, December 79, pp. 397-434.
- [COU_72] Coulouris, G. H.; Evans, J. M.; Mitchell, R. W. - "Towards Content Addressing in Databases", Computer Journal, Vol. 15., Nro. 2, May 72, pp. 95-98.
- [DAT_75] Date, C. J. - "An Introduction to Database Systems" Addison-Wesley Publishing Company, 1975.
- [DEB_85] Debenham, J. K. - "Knowledge Base Design" Australian Computer Journal, Vol. 17, Nro. 1, February 85, pp. 42-48.
- [DEM_85] Demurjian, S. A. et alli - "Performance Evaluation of a Database System in Multiple Backend Configurations", in Proc. 4th Int. Workshop on Database Machines, March 1985, pp. 91-111.
- [DEW_79] DeWitt, D. J. - "Direct - A Multiprocessor Organization for Supporting Relational Database

Management Systems", IEEE Trans. on Computers, Vol. C-28, Nro. 6, June 1979, pp.395-406.

- [DIG_79a] "CP/M 2 Interface Guide", Digital Research, Pacific Grove, California, 1979.
- [DIG_79b] "CP/M 2 Alteration Guide", Digital Research, Pacific Grove, California, 1979.
- [EAR_85] Earl, A. N.; Whittington, R. P. - "Capturing the Semantics of an IPSE Database", Data Processing, Vol.27, Nro. 9, November 1985, pp. 33-43.
- [ENC_80] Encarnação, J.; Neumann, T. - "A Survey of DB Requirements for Graphical Applications in Engineering", Lecture Notes in Computing Science, Vol. 81, New York: Springer-Verlag, 1980, pp. 285-297.
- [ESW_76] Eswaran, K. P. et al. - "The Notion of Consistency and Predicate Locks in a Database System" Commun. ACM, Vol. 19, Nro 11, November 76, pp. 624-633.
- [FAR_85] Farmer, D. B.; King, R.; Myers, D. A. - "The Semantics Database Constructor" IEEE Trans. Software Engineering, Vol. SE-11 Nro. 7, July 85, pp. 583-591.
- [FER_75] Ferreira, A. B. H. - "Novo Dicionário da Língua Portuguesa" Editora Nova Fronteira, 1.a Edição 1975.
- [FOA-85] Foard, R. M. - "A Data Manager Using Entity-Relationships", PC Tech Journal, Vol. 3 Nro. 10, October 85, pp. 96-126.
- [FOR_85] Ford, R.; Miller, K. - "Abstract Data Type development and Implementation: An Exemple", IEEE Trans. Software Engineering, Vol. SE-11 Nro. 10, October 85, pp. 1033-1037.
- [GAN_79] Gane, C.; Sarson, T. - "Análise Estruturada de Sistemas", Livros Técnicos e Científicos, 1.a Edição, 1983.
- [GAR_85] Garner, B. J.; Tsui, E. - "Knowledge Representation for an Audit Office", Australian Computer Journal, Vol. 17, Nro. 3, August 85, pp. 106-112.
- [GRA_78] Gray, J. - "Notes on Database Operating Systems", Lecture Notes in Computing Science, Vol. 60, New York: Springer-Verlag, 1978, pp. 394-481.
- [GUT_77] Guttag, J. J. - "Abstract Data Types and the Development of Data Structures", Commun. ACM, Vol. 20 Nro. 6, June 1977, pp. 396-404.
- [GUT_78a] Guttag, J. V.; Horning, J. J. - "The Algebraic Specification of Abstract Data Types", Acta Inform. Vol. 10, 1978, pp. 27-52.

- [HAD_84] Haderle, D. J.; Jackson, R. D. - "IBM Database 2 Overview", IBM Systems Journal, Vol. 23, Nro. 2, 1984, pp. 112-125.
- [HAH_85] Hahne, K.; Pilgram, P.; Schuett, D.; Schweppe, H.; Wollf, G. - "Associative Processing in Standard and Deductive Databases", in Proc. 4th Int. Workshop on Database Machines, March 1985, pp. 1-12.
- [HAM_81] Hammer, M.; McLeod, D. - "Database Description with SDM: A Semantic Database Model", ACM Trans. Database Systems, Vol. 6, Nro. 3, September 81, pp. 351-386.
- [HAR_85] Hartzband, D. J.; Maryanski, F. J. - "Enhancing Knowledge Representation in Engineering Databases", IEEE Computer, Vol. 18, Nro. 9, September 85, pp. 39-48.
- [HAW_82] Hawthorn, P. B.; DeWitt, D. J. - "Performance Analysis of Alternative Database Machine Architecture", IEEE Trans. on Software Engineering, Vol. SE-8, Nro.1, January 1982, pp. 61-75.
- [HIK_85] Hikita, S.; Kawakami, S.; Haniuda, H. - "Database Machine FRIEND", in Proc. 4th Int. Workshop on Database Machines, March 1985, pp. 190-207.
- [HIL_86] Hillyer, B. K.; Shaw, D. E.; Nigam, A. - "NON-VONs Performance on Certain Database Benchmarks", IEEE Trans. on Software Engineering, Vol. SE-12, Nro. 4, April 1986, pp. 577-583.
- [HDN_85] Hong, Y. C. - "Efficient Computing of Relational Algebraic Primitives in a Database Machine Architecture", IEEE Trans. on Computers, Vol. C-34, Nro. 7, July 85, pp. 588-595.
- [HRR_69] Harary, F. - "Graph Theory", Addison-Wesley, 1969.
- [JAJ_83] Jajodia, S.; Ng, P. A.; Springsteel, R. N. - "The Problem of Equivalence for Entity Relationship Diagrams", IEEE Trans. Computers, Vol. SE-9, Nro. 5, September 83, pp. 617-630.
- [KAK_85] Kakuta, T.; Miyasaki, N.; Shibayama, S.; Yokota, H.; Murakami, K. - "The Design and Implementation of Relational Database Machine Delta", in Proc. 4th Int. Workshop on Database Machines, March 1985, pp.
- [KAR_81] Karlsson, K. - "Reduced Cover-trees and their Application in the SABRE Access Path Model", in proc. 7th Int. Conference on Very Large Data Bases, Sept. 81.
- [KAT_84] Katz, R. H.; Lehman, T. J. - "Database Support for Versions and Alternatives of Large Design Files", IEEE Trans. Computers, Vol. SE-10, Nro. 2, march 84, pp. 191-200.
- [KEL_86] Keller, A. M. - "The Role of Semantics in

Translating View Updates", IEEE Computer, Vol. 19, Nro. 1, January 86, pp. 34-49.

- [KGK_84] Kim, W.; Gajski, D.; Kuck, D. J. - "A Paralell Pipelined Relational Query Processor", ACM Trans. on Database Systems, Vol. 9, Nro. 2, June 84, pp. 214-242.
- [KLG_86] Kellogg, C. - "From Data management to Knowledge Management" IEEE Computer, Vol. 19, Nro. 1, January 86, pp. 75-84.
- [KRU_83] Kruglinski, D. - "Data Base Management Systems - A Guide to Microcomputer Software", Osborne/McGraw-Hill, Berkeley, California, 1983.
- [KUL_86] Kulkarni, K.G.; Atkinson, M. P. - "EFDM: Extended Functional Model", The Computer Journal, Vol. 29, Nro. 1, 1986, pp. 38-46.
- [LAN_78] Langdon, G. G. - "A Note on Associative Processors for Data Management", ACM TODS, Vol3, June 1978, pp. 148-158.
- [LEA_83] Leath, C. L.; Ollanik, S. J. - "Software Architecture for the Implementation of a Computer Aided Engineering System", in Proc. 20th Design Automation Conference, June 83, pp. 137-142.
- [LEL_85] Leland, M. D. P.; Roome, W. D. - "The Silicon Database Machine", in Proc. 4th Int. Workshop on Database Machines, March 1985, pp. 169-189.
- [LIN_85] Lin, K.; Gannon, J. D. - "Atomic Remote Procedure Call", IEEE Trans. on Software Engineering, Vol. SE-11, Nro. 10, October 85, pp. 1126-1135.
- [LSS_76] Lin, S. C., Smith, D. C. P., Smith, J. M. - "The Design of a Rotational Associative Memory for Relational Database Applications", ACM TODS, Vol. 1, March 76, pp. 53-75.
- [MAR_83] Marshall, T. G.; Attikiouzel, J. A. - "Floppy Disk Data Transfer Techniques", IEEE Micro, Vol. 3, Nro. 6, December 1983, pp. 17-23.
- [MAT_81] Matsuka, H.; Uno, S.; Sibuya, M. - "Specific Requirements in Engineering Data Base", Lecture Notes in Computing Science, Vol. 133, New York: Springer-Verlag, 1982, pp. 345-356.
- [NAV_86a] Navathe, S. et al. "Integrating User Views in Database Design", IEEE Computer, Vol. 19, Nro. 1, January 86, pp. 50-62.
- [NAV_86b] Navathe, S.; Kerschberg, L. - "Role of Data Dictionaries in Information Resource Management", Information and Management, Vol. 10, 1986, pp. 21-46.

- [NG*_81] Ng, A. P. - "Further Analysis of the Entity-Relationship Approach to Database Design", IEEE Trans. Software Engineering Vol. SE-7 Nro. 1, January 81, pp. 85-99.
- [OZK_77] Ozkarahan, E. A.; Schuster, S. A.; Sevcik, K. C. - "Performance Evaluation of a Relational Associative Processor", ACM TODS, Vol. 2., June 77.
- [PAR_85] Parent, C.; Spaccapietra, S. - "An Algebra for a General Entity-Relationship Model", IEEE Trans. Software Engineering, Vol. SE-11 Nro. 7, July 85, pp. 634-643.
- [PET_77] Peterson, J. L. - "Petri Nets", ACM Computing Surveys, Vol. 9, Nro. 3, September 77, pp. 223-252.
- [QAD_85a] Qadah, G. Z.; Irani, K. B. - "A Database Machine for Very Large Relational Databases", IEEE Trans. on Computers, Vol. C-34, Nro. 11, November 1985, pp. 1015-1026.
- [QAD_85b] Qadah, G. Z. - "The Equi-Join Operation on a Multiprocessor Database Machine: Algorithms and the Evaluation of their Performance", in Proc. 4th Int. Workshop on Database Machines, March 1985, pp. 35-67.
- [RIE_77] Ries, D. R.; Stonebraker, M. - "Effects of Locking Granularity in a Database Management System", revisited", ACM Trans. Database Systems, Vol. 2, Nro. 3, September 77, pp. 233-246.
- [RIE_79] Ries, D. R.; Stonebraker, M. - "Locking Granularity revisited", ACM Trans. Database Systems, Vol. 4, Nro. 2, June 79, pp. 210-227.
- [SAK_84] Sakai, H. et alli - "Design and Implementation of the Relational Database Engine", in Proc. Int. Conference on 5th Generation Computer Systems, 1984, pp. 419-426.
- [SAR_85] Sarin, S. K.; Blaustein, B. T.; Kaufman, C. W. - "System Architecture for Partition Tolerant Distributed Databases", IEEE Trans. Computers, Vol. C-34, Nro. 12, December 85, pp. 1158-1163.
- [SBY_84] Shibayama, S. et alli - "Query Processing Flow on RDBM Delta's Functionally-Distributed Architecture", in Proc. Int. Conference on 5th Generation Computer Systems, 1984, pp. 427-435.
- [SCH_79] Schuster, S. A.; Nguyen, H. B., Ozkarahan, E. A.; Smith, K.C. - "RAP.2 - An Associative Processor for Databases and Its Applications", IEEE Trans. on Computers, Vol. C-28, Nro. 6, June 1979, pp. 446-458.
- [SHE-85] Sheth, A. P. et al. - "An Analysis of the Effect of Network Parameters on the Performance of Distributed Database Systems", IEEE Trans. on

- [SHI_81] Shipman, D. W. - "The Functional Data Model and the Data Language DAPLEX", ACM Trans. Database Systems, Vol. 6, Nro. 1, March 81, pp. 140-173.
- [SHO_85] Shoshani, A.; Wong, K. T. - "Statistical and Scientific Database Issues", IEEE Trans. on Software Engineering, Vol. SE-11, Nro. 10, October 85, pp. 1040-1047.
- [SHR_82] Shrivastava, S. K.; Panzieri, F. - "The Design of a Reliable Remote Procedure Call Mechanism", IEEE Trans. on Computers, Vol. C-31, Nro. 7, July 82, pp. 692-697.
- [SHU_84] Shultz, R. K.; Zingg, R. J. - "Response Time Analysis of Multiprocessor Computers for Database Support", ACM Trans. on Database Systems, Vol. 9, Nro. 1, March 84, pp. 100-132.
- [SIM_83] Simon, R. L. - "CAD/CAM - The Foundation for Computer Integrated Manufacturing", in Proc. 20th Design Automation Conference, June 83, pp. 686-700.
- [SLO_70] Slotnik, D. L. - "Logic per Track Devices", in Advancing in Computers, Vol. 10. F. Alt. Ed. New York: Academic, 1970, pp. 291, 296.
- [SMI_77a] Smith, J. M.; Smith, D. C. P. - "Database Abstractions: Aggregation", Commun. ACM, Vol. 20, Nro. 6, June 1977, pp. 405-413.
- [SMI_77b] Smith, J. M.; Smith, D. C. P. - "Database Abstractions: Aggregation and Generalization", ACM Trans. Database Systems, Vol. 2, Nro. 2, June 77, pp. 105-133.
- [SSU_78] SU, S. Y. W.; Emam, A. - "CASDAL: CASSM's Data Language", ACM Trans. on Database Systems, Vol. 3, Nro. 1, March 78, pp. 57-91.
- [STD_76] Stonebraker, M. E. et alli - "The Design and Implementation of INGRES", ACM Trans. Database Systems, Vol. 1, Nro. 3, September 76, pp. 189-222.
- [TAK_84] Taki, K. et alli - "Hardware Design and Implementation of the Personal Sequential Inference Machine (PSI)", in Proc. Int. Conference on 5th Generation Computer Systems, 1984, pp. 398-409.
- [TEI_77] Teichroew, D.; Hershey III, E. A. - "PSL/PSA: A Computer Aided Technique for Structured Documentation and Analysis of Information Processing Systems", IEEE Trans. Software Engineering Vol. SE-3 Nro. 1, January 77, pp. 41-48.
- [TRA_76a] Traina Jr., C. - "Manual de Uso do Sistema PSL/PSA 9830A", SCE Documento de Trabalho, São Carlos,

1976.

- [TRA_76b] Traina Jr., C. - "Manual de Referência do Sistema PSL/PSA 9830A", SCE Documento de Trabalho, São Carlos, 1976.
- [TRA_77] Traina Jr., C. - "Estudo Comparativo Entre Possíveis Representações Feitas na Linguagem PSL", SCE Documento de Trabalho, São Carlos, 1977.
- [TRA_78] Traina Jr., C.; Germano, F. S. R.; Sahão Jr., J. - "Uso de Dígrafos para Esquematização de uma Base de Dados", In Anais do 2º Congresso Brasileiro de Automática, Florianópolis, SC, Setembro de 1978.
- [TRA_82] Traina Jr., C. - "Sistema de Apoio por Computador à Documentação de Sistemas", Dissertação apresentada ao ICMSC-USP para obtenção do Título de Mestre, Dezembro 82.
- [TRA_85] Traina Jr., C. et alli - "Gerenciamento de Configuração de Software Usando o Modelo Entidade-Relacionamento", in Anais do 2º Congresso Nacional de Automação Industrial, São Paulo, Novembro 85, pp. 387-395.
- [TRA_86] Traina Jr., C. et alli - "SIPS - Estado Atual de Desenvolvimento", in Anais do 19º Congresso Nacional de Informática, Rio de Janeiro, Agosto de 1986.
- [TSU_85] Tsukumo, A. N. et alli - "Um Sistema Expansível para Produção de Software", in Anais do 2º Congresso Nacional de Automação Industrial, São Paulo, Novembro 85, pp. 379-386.
- [VAL_84] Valduriez, P.; Gardarin, G. - "Join and Semijoin Algorithms for a Multiprocessor Database Machine", ACM Trans. on Database Systems, Vol. 9, Nro. 1, March 84, pp. 133-161.
- [WSU_79] Su, S. Y. W.; Nguyen, L. H.; Emam, A.; Lipovski, G. J.; - "The Architectural Features and Implementation Techniques of the Multicell CASSM", IEEE Trans. on Computers, Vol. C-28, Nro. 6, June 1979, pp. 430-445.
- [WSU_86] Su, S. Y. W. - "Modeling Integrated Manufacturing Data with SAM*", IEEE Computer, Vol. 19, Nro. 1, January 86, pp. 34-49.
- [YOK_84] Yokota, M. et alli - "A Microprogrammed Interpreter for the Personal Sequential Inference Machine", in Proc. Int. Conference on 5th Generation Computer Systems, 1984, pp. 410-418.