ASPECTOS DE PARALELISMO NA GERÊNCIA DE DADOS E OBJETOS NO GEOTABA

Marta Lima de Queirós Mattoso

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO EM ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Jano Moreira de Souza, Ph.D.
(Presidente)

Profa. Ana Regina C. da Rocha, D.Sc.

Prof. Cláudio Luis de Amorim, Ph.D.

Profa. Ana Maria de Carvalho Moura, D.Ing.

Prof. Rubens Nascimento Melo, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
ABRIL DE 1993

MATTOSO, MARTA LIMA DE QUEIRÓS

Aspectos de Paralelismo na Gerência de Dados e Objetos do GEOTABA [Rio de Janeiro] 1993.

ix, 100p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 1993)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. banco de dados 2. orientação a objetos 3. processamento paralelo 4. arquiteturas de banco de dados

I. COPPE/UFRJ II. Título (série)

À Beta,

por sua imensa bondade

e total desprendimento

Agradecimentos,

Ao Professor Jano Moreira de Souza, pela orientação, carinho e incentivo pelo trabalho, além de sua dedicação ao promover condições favoráveis ao desenvolvimento deste trabalho.

À Professora Ana Regina Cavalcanti da Rocha, por ter sempre um gesto incentivador mesmo nos momentos mais difíceis.

Ao Professor Cláudio Luis de Amorim por seu apoio e incentivo ao processamento paralelo e seu empenho na liderança do projeto NCP.

Aos companheiros e ex-companheiros do Projeto TABA, Luiz Carlos, Cláudia, Trotta, Teresa, Guilherme, Lygia, Gilda, Neide e Vera, pelos momentos alegres e pelo companheirismo nas horas mais difíceis.

Ao chefe do laboratório, durante a fase de implementação deste trabalho, Eliseu Chaves por sua dedicação e imensa disposição para ajudar.

Ao Edson e Cláudia Andréia pelo apoio na implementação do servidor de objetos seqüencial.

Aos demais professores, colegas e funcionários da COPPE / Sistemas, em especial às Profas. Lidia Segre e Sueli Mendes, à Denise Cupollilo e Cláudia Prata, que sempre me incentivaram, colaborando ao longo da evolução deste trabalho.

Ao Professor Paulo Roberto de Oliveira, pelo primeiro incentivo à pesquisa através da orientação em Projetos de Iniciação Científica.

Ao Professor Nelson Francisco Favilla Ebecken, por sua apreciação deste trabalho, colaborando com a fase final desta empreitada.

Aos meus pais e irmãs por seu carinho eterno e pela convivência harmoniosa que proporcionam.

Aos meus filhos Liddy e André por seu amor, alegria e espontaneidade que muito me ajudaram.

Finalmente, ao meu grande amor Álvaro, pelo eterno apoio, incentivo, compreensão e por seu amor multíplice.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências (D.Sc.).

ASPECTOS DE PARALELISMO NA GERÊNCIA DE DADOS E OBJETOS NO GEOTABA

Marta Lima de Queirós Mattoso

ABRIL, 1993

Orientador: Jano Moreira de Souza

Programa: Engenharia de Sistemas e Computação

Este trabalho se insere no contexto do GEOTABA, sistema de gerência de objetos da Estação de Trabalho TABA. Essa Estação de Trabalho visa dar suporte ao engenheiro de software nas diversas fases do ciclo de vida de um software, incluindo a configuração e implementação de ambientes adequados ao seu desenvolvimento em diversas áreas de aplicação. Os habitantes da estação TABA se enquadram na classe de aplicações não convencionais para um SGBD, surgindo então a motivação do desenvolvimento de um sistema de gerência de objetos para centralizar o controle dos objetos da estação TABA.

Considerando-se a necessidade de desempenho que as aplicações de ambientes de desenvolvimento de software impõem, esta tese dedica-se ao estudo e desenvolvimento de várias técnicas que contribuem para aumentar o desempenho da gerência de dados e objetos do GEOTABA. Desta forma, foram utilizadas tecnologias novas no contexto de Sistemas de Banco de Dados, a saber: i) processamento paralelo, ii) orientação a objetos e iii) comunicação cliente/servidor. Além do uso dessas três tecnologias ainda não estar solidificado no contexto de banco de dados, a sua combinação tornou este trabalho uma tarefa desafiante.

Nesse sentido, três protótipos foram desenvolvidos com sucesso: i) o servidor paralelo de consultas relacionais - PARBASE, ii) o servidor sequencial de objetos - GOA e, iii) o servidor paralelo de objetos - PARGOA. Os resultados obtidos com os diversos experimentos realizados, confirmam a presença do processamento paralelo no futuro dos Sistemas de Bancos de Dados de Alto Desempenho.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.).

PARALLEL ASPECTS IN THE DATA AND OBJECT MANAGEMENT OF GEOTABA

Marta Lima de Queirós Mattoso

APRIL, 1993

Thesis Supervisor: Jano Moreira de Souza

Department: Systems and Computer Engineering

This work is part of GEOTABA - the Object Management System of TABA Workstation. This Workstation intends to support software engineering during the phases of a software lifecycle, including the configuration and implementation of suitable environments in several application fields. The components of TABA Workstation falls into the category of non conventional applications for the Database Management System standpoint.

This non conventional environment provides the main motivation for the development of an Object Management System for TABA Workstation. Since 'CASE' - Computer Aided Software Engineering, applications require high performance, this thesis is centered on the development and study of several performance enhancement techniques for Objetc and Data Management. Those include: i) parallel processing, ii) object orientation and iii) client/server commuincation. Each of these techniques are under intensive research with the database community, and their blending might be considered a challenging task.

Within this framework, three prototypes were successfuly developed and implemented: i) PARBASE - a parallel relational query server, ii) GOA - an object sequential server, and iii) PARGOA - a parallel object server. Performance measurements were undertaken on the NCP I - a hypercube machine developed at the Computer Science Department of COPPE/UFRJ, where good speed-up's were obtained for many different types of queries. These results have ensured the presence of parallel processing in the future of high performance database systems.

ÍNDICE

Capítulo I - INTRODUÇÃO

I.1 Motivação: O Projeto TABA 1

I.2 Objetivo da Tese 2

I.3 Trabalho Realizado 4

Capítulo II - BANCOS DE DADOS E PARALELISMO

II.1 Introdução 6

II.2 Caracterização de SBDs Paralelos 7

II.2.1 Acoplamento ao hardware 7

II.2.2 Parâmetros da arquitetura hospedeira 8

II.2.2.1 Modelos de memória 8

II.2.3 Acoplamento ao sistema operacional 9

II.2.4 Carregamento do código 10

II.2.5 Particionamento dos dados 10

II.2.6 Nível de granularidade do paralelismo 11

II.2.7 Modelo de dados 12

II.2.8 Algoritmos das operações 12

II.3 Sistemas de Banco de Dados Paralelos Atuais 13

II.4 Comentários Finais 16

Capítulo III - O Servidor Paralelo de Consultas Relacionais

III.1 Preliminares 17

III.2 O Ambiente de Desenvolvimento do PARBASE 17

III.2.1 Ambiente computacional 18

III.2.2 Ambiente de programação STRAND 19

III.3 O Protótipo PARBASE 20

III.3.1 Características do protótipo PARBASE 20
III.3.2 A implementação do PARBASE 21
III.4 Análise do Desempenho do PARBASE 23
III.4.1 Seleção 24
III.4.2 Junção 27
III.5 Comentários sobre a Experiência Realizada 28

Capítulo IV - A Arquitetura do SGO GEOTABA

IV.1 Preliminares 29
IV.2 Tecnologias de arquiteturas de SGBDOOs 30
IV.2.1 Plataformas Cliente/Servidor em arquiteturas de SGBDOOs 32
IV.3 A arquitetura do GEOTABA 35
IV.3.1 A arquitetura funcional do GEOTABA 35
IV.3.2 O GEOTABA na arquitetura cliente/servidor 37
IV.3.3 Aspectos de implementação da arquitetura do GEOTABA 40
IV.3.4 Os modos de operação da arquitetura do GEOTABA 44
IV.3.5 Características do Projeto GEOTABA 45

Capítulo V - O Servidor do Gerente de Objetos Armazenados

V.1 Preliminares 46
V.2 A Gerência do Espaço de Armazenamento 46
V.3 Estruturas de Armazenamento 47
V.4 As Operações do Servidor 52
V.5 O Avaliador de Consultas 53
V.5.1 Alternativas de Projeto do Avaliador de Consultas 54
V.5.2 Escopo e Resultados da Consulta 54
V.5.3 A Pseudo-Sintaxe da Linguagem de Consulta 55
V.5.4 A Implementação do Processador de Consultas 57

V.5.5 Avaliação de Estratégias de Processamento de Consultas 59

V.5.5.1 Avaliação da Implementação da Estratégia Ascendente 61

Capítulo VI - O Servidor Paralelo do SGO GEOTABA

VI.1 Preliminares 64

VI.2 O Paralelismo e a Orientação a Objetos 65

VI.2.1 O Uso de Linguagens de Programação Paralelas Orientadas a Objetos 66

VI.2.2 O Paralelismo nas Operações do SGBDOO 67

VI.2.3 A Questão do Agrupamento X Particionamento 68

VI.2.4 As Soluções Anteriores 68

VI.3 A Solução Adotada na Gerência Paralela de Objetos 70

VI.3.1 Características da Arquitetura de Disco Compartilhado 70

VI.3.2 Comparando os Diversos Modelos de Memória 71

VI.4 O Paralelismo na Arquitetura do GEOTABA 72

VI.5 O Protótipo PARGOA 74

VI.5.1 O Ambiente de Desenvolvimento 75

VI.5.2 Características do PARGOA 76

VI.5.3 A Implementação do PARGOA 78

VI.5.4 Análise das Operações de Consulta 80

Capítulo VII - Conclusões 85

Referências Bibliográficas 89

Capítulo I

INTRODUÇÃO

I.1 MOTIVAÇÃO: O PROJETO TABA

Ambientes de desenvolvimento de software (ADSs) vêm sendo propostos e utilizados como apoio às diversas fases do ciclo de vida de um software. À medida que novas classes de aplicações fazem uso de ferramentas específicas que suportam métodos adequados ao desenvolvimento de seu software, surge a necessidade da utilização de ambientes para desenvolvimento de software especialmente configurados para esse propósito. Nesse sentido, o Projeto TABA [Roch90], em andamento no Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ, visa prover o engenheiro de software com uma Estação de Trabalho que possa ser configurada por um especialista, com o auxílio de técnicas de inteligência artificial, para desenvolver um projeto específico, levando em conta características da equipe desenvolvedora, do tipo de aplicação, do perfil do usuário, custos e grau de confiabilidade desejados, etc. [Agui92].

A variação de ambientes de desenvolvimento de software encontrados, seja através de produtos, protótipos ou através de propostas na literatura, é muito vasta e diversos autores [Dart87, Pene88 , Perry88, Mour92a] se preocuparam em propor classificações, taxonomias e modelos no sentido de caracterizar e comparar esses ambientes. Nesse sentido, foi realizada uma caracterização do TABA em [Matt92]. Esse trabalho comenta algumas taxonomias encontradas na literatura, discute algumas definições e utiliza a classificação de Penedo [Pene88] para caracterizar o TABA.

Um sistema de gerência de bases de dados (SGBD) surge como componente central na arquitetura da estação de trabalho TABA. É através do SGBD que as ferramentas serão integradas e o seu uso controlado. Entretanto, os SGBDs convencionais não têm se mostrado adequados no suporte aos dados que trafegam num ambiente de desenvolvimento de software [Souz91].

A literatura e os protótipos existentes apontam basicamente duas soluções para a gerência de dados não convencionais: os sistemas de banco de dados orientadods a objetos (SGBDOOs) [Atki89] e os sistemas que estendem o modelo relacional para suportar a gerência de dados não convencionais [Ston89]. O sucesso de protótipos/produtos de SGBDOOs como O_2 [Deux91], Orion (Itasca) [Kim 90b], GemStone [Butt91], VBase [Catt91], entre outros, aliados à potencialidade dos conceitos da orientação a objetos para o desenvolvimento de um modelo de dados rico em semântica, levaram à escolha do paradigma da orientação a objetos na gerência de

dados do projeto TABA. Mais especificamente, para aplicações de engenharia de software são encontradas na literatura [Heil87, Bern87, Huds87] uma série de requisitos e características de ambientes de desenvolvimento de software que recomendam a utilização do paradigma da orientação para objetos como suporte à modelagem dos diversos tipos de dados existentes num ADS.

Existe uma série de modelos de dados que serão utilizados pelas diversas ferramentas da estação TABA em suas diversas interfaces. Surge então a necessidade da especificação de um modelo de objetos para o qual todos os modelos de dados da estação sejam mapeados. Como conseqüência direta, torna-se necessário o desenvolvimento de um sistema de gerência de objetos(SGO) que suporte diretamente este modelo de objetos e que centralize o controle dos objetos da estação TABA de modo eficiente.

É no contexto do sistema de gerência de objetos da estação TABA - o GEOTABA [Matt89], que surge a motivação do trabalho desta tese, no sentido de apresentar técnicas de Banco de Dados adequadas ao ambiente TABA.

I.2 OBJETIVO DA TESE

As arquiteturas de SGBDs tradicionalmente se preocupam com o desempenho no acesso aos dados da aplicação através do uso de técnicas para persistência dos dados independente do programa de aplicação, da utilização de linguagens eficientes de consulta de alto nível, além de técnicas otimizadas para a gerência da memória secundária e de transações. Segundo Bancilhon [Banc92] essa ênfase da tecnologia de Bancos de Dados em desempenho tornou a sua integração com a programação orientada a objetos uma tarefa desafiadora.

Diversos autores [Kim 90a, DeWi90c, Catt91] apontam dois aspectos como entraves para a aceitação dos SGBDOOs por seus potenciais usários, a saber: a **padronização** e o **desempenho**. A falta de padronização e a aceitação de um modelo básico que sirva de referência para SGBDOOs, leva a comunidade de usuários a uma confusão de termos e conceitos gerando uma relutância em relação ao abandono de padrões estabelecidos, como a SQL por exemplo. Vários esforços vêm sendo realizados no sentido da obtenção de uma padronização através de grupos de trabalho com organizações como a ANSI por exemplo [Kim90a, Catt91].

Já o desempenho, devido à demanda de tempo de processamento das novas aplicações não convencionais, aliado às novas características dos dados modelados através da orientação a objetos levam os projetistas de SGBDOOs à preocupação com a otimização do tempo de resposta. A atenção dos projetistas se volta, então, para a exploração de novas técnicas de projeto de arquiteturas voltadas para a otimização do tempo de resposta dos SGBDOOs [Catt91].

No caso da aplicação em ADS (Ambiente de Desenvolvimento de Software) para SGBDOOs [Catt91, DeWi90c], e especificamente no Projeto TABA [Matt92], a manipulação dos dados tem como características:

- ⇨ grande parte da aplicação (ferramentas de desenvolvimento de software) encontra-se nos métodos das classes onde os objetos são instâncias;
- ⇨ a necessidade de consultas associativas sobre conjuntos de objetos (por exemplo, ferramentas de apoio a programação que necessitam encontrar as referências a determinada sub-rotina ou variável); e
- ⇨ a necessidade de interfaces gráficas e capacidade computacional que somente uma estação de trabalho dedicada pode suportar.

Desta forma, uma solução de SGBDOO centralizada não é mais aceitável, surgindo a necessidade de uma arquitetura de SGBDOO distribuída, compatível com os requisitos de desempenho da aplicação. No âmbito de novas arquiteturas distribuídas para SGBDOOs aparece a oportunidade de exploração dos computadores paralelos disponíveis no mercado. Em particular, na COPPE está disponível para experimentos um protótipo do computador paralelo Macaco - Máquina de Cálculo Coordenado[Amor91] contando com oito processadores ligados em topologia hipercúbica - o NCP I (totalmente desenvolvido pelos pesquisadores da linha de pesquisa de arquiteturas de computadores do Programa de Engenharia de Sistemas da COPPE/UFRJ). O NCP I oferece a oportunidade de utilização de uma máquina paralela com sistema operacional compatível com Unix, com a possibilidade de comunicação com estações de trabalho Sun.

O objetivo desta tese consiste em contribuir para um melhor desempenho na gerência dos dados do Projeto TABA, através do projeto de uma nova arquitetura para SGBDOOs validada com experiências para gerência distribuída e paralela de dados em plataformas cliente/servidor com hardware paralelo.

Observa-se que as soluções para o problema de desempenho encontradas na maioria dos protótipos/produtos de SGBDOOs atuais, preocupam-se exclusivamente com o aumento de desempenho para dados convencionais ou com o suporte à gerência de dados não convencionais. Nessas soluções, são encontradas respectivamente, as seguintes tecnologias:

- ⇨ Bancos de dados relacionais para máquinas paralelas, solução com grande aumento de desempenho, mas sem suportar objetos complexos.
- ⇨ Arquiteturas de sistemas de gerência de objetos com a plataforma cliente/servidor. Esta solução suporta a gerência de objetos complexos e explora o potencial de diversas UCPs de estações de trabalho no paradigma seqüencial.

A solução proposta faz a união das soluções anteriores, através do projeto de uma arquitetura cliente/servidor com o servidor de objetos paralelo. A tecnologia de processamento paralelo de dados convencionais é adaptada para o suporte à gerência de objetos em paralelo no contexto da arquitetura do GEOTABA.

O uso de paralelismo na arquitetura do SGBDOO GEOTABA é também incentivado pelo crescente aparecimento de multiprocessadores no parque computacional atual, seja através de estações de trabalho com vários processadores, ou seja através de computadores paralelos de custo equivalente ao de estações de trabalho. À medida que as máquinas paralelas vão surgindo, seus custos diminuem e a utilização de uma máquina paralela no lugar de uma estação de trabalho seqüencial não será impedida pelo fator custo, mas talvez pela ausência de tecnologia para o novo ambiente. Desta forma, este trabalho une a necessidade de desempenho das aplicações não convencionais ao potencial do processamento paralelo.

I.3 TRABALHO REALIZADO

Para atingir o objetivo desejado, foram necessários estudos e experiências no sentido de dominar as tecnologias avançadas de processamento paralelo, orientação a objetos e arquiteturas cliente/servidor. A seguir são descritos os trabalhos realizados nessa direção.

Inicialmente foi elaborado junto à equipe do Projeto TABA, um estudo preliminar sobre gerência e armazenameneto de objetos. Foi realizado um levantamento e estudo comparativo de diversos protótipos de sistemas de gerência de objetos encontrados na literatura. Essas análises enfocaram os diversos componentes de um SGO e estão disponíveis na forma de relatórios técnicos [Gonç88, Degr88, Ross89] e não serão objeto de apresentação no escopo do presente trabalho.

Em seguida foi realizada a revisão da literatura na tecnologia de uso de paralelismo em Banco de Dados que ganhou um capítulo a parte, o **capítulo II**, por se tratar de um tema recente de pesquisa e ainda não coberto em livros didáticos à excessão de apenas um capítulo em Oszü e Valduriez [Oszü90]. O capítulo II também apresenta uma análise de sistemas de gerência de banco de dados paralelos com algumas considerações quanto à utilização de paralelismo na orientação a objetos.

No sentido de avaliar técnicas de paralelismo em plataformas reais, foi realizada a experiência de implementação do PARBASE - um Banco de Dados Paralelo com o modelo relacional de dados no NCP I [Amor91]. São apresentadas as avaliações de desempenho mostrando o ganho obtido, além da descrição do desenvolvimento do próprio PARBASE no **capítulo III**.

Para explorar paralelismo no contexto da orientação a objetos, foi necessária a especificação de uma arquitetura de sistema de gerência de objetos que comportasse o uso de paralelismo no contexto do GEOTABA. Foram estudadas técnicas de orientação a objetos para gerência de objetos e processamento de consultas. Essas técnicas foram avaliadas e exploradas através da implementação do GOA - o Gerente de Objetos Armazenados do GEOTABA com capacidade de avaliação de predicados de consulta. O desenvolvimento do GOA permitiu que diferentes estratégias de resolução de consultas fossem exploradas. O **capítulo IV** faz uma breve apresentação sobre as tecnologias envolvidas na construção de SGBDOOs e descreve a arquitetura do GEOTABA. As experiências do desenvolvimento do GOA, encontram-se descritas no **capítulo V** .

Finalmente, as experiências com as implementações dos algoritmos paralelos do PARBASE, aliadas à implementação do servidor GOA e seu processador de consultas, contribuiram para o projeto de um servidor de objetos adequado à exploração de técnicas de paralelismo. Foram estudadas as dificuldades encontradas na paralelização de SGBDOOs. Este trabalho especifica e implementa o PARGOA - Servidor Paralelo de Objetos, na máquina NCP I, onde são utilizados algoritmos paralelos para a gerência/manipulação de objetos armazenados e não de relações. No PARGOA, são suportadas todas as operações do GOA de gerência de objetos e processamento de consultas, através do mesmo modelo físico de armazenamento dos objetos. Não foram encontrados na literatura, trabalhos que utilizassem técnicas de paralelismo na manipulação de objetos. Alguns projetos encontrados apresentam a proposta de SGBDs com modelos de dados orientados a objetos, mas que, no entanto, fazem o mapeamento dos dados para o modelo relacional no nível físico de representação. Nesses projetos, a solução de paralelismo é aplicada sobre o modelo relacional de representação de dados. O **capítulo VI** apresenta a exploração do paralelismo dentro da orientação a objetos, através de discussões sobre os problemas encontrados e através da descrição da implementação do servidor paralelo de objetos PARGOA no contexto da arquitetura do GEOTABA apresentando suas avaliações de desempenho mostrando o ganho obtido.

O **capítulo VII,** encerra este trabalho apresentando os comentários conclusivos sobre as contribuições das experiências realizadas e apontando as perspectivas futuras de continuidade na exploração de paralelismo em sistemas de gerência de bases de dados orientados a objetos.

Capítulo II

BANCOS DE DADOS E PARALELISMO

II.1 INTRODUÇÃO

O avanço da nova tecnologia de arquiteturas de máquinas paralelas, está trazendo benefícios para diversas áreas de aplicação em computação e entre elas está a de Banco de Dados. O objetivo da utilização de arquiteturas paralelas na hospedagem de um Sistema de Banco de Dados (SBD) está em se atingir o máximo de paralelismo durante a interação do usuário com o SBD visando o aumento do desempenho. A habilidade de se explorar um grande número de processadores conectados em paralelo é hoje uma área de pesquisa intensa.

O desempenho de aplicações de banco de dados em máquinas seqüenciais vem se tornando um fator crítico para aplicações que envolvem consultas complexas e relações com grande volume de dados. O processamento de operações de banco de dados oferece várias oportunidades de paralelismo que apontam para um ganho no tempo de resposta dos SBDs. Entretanto, o ganho geral em desempenho foi muito questionado até recentemente devido ao estrangulamento na entrada/saída de dados. Observa-se, no entanto, que evoluções no hardware e no software têm mascarado parcialmente este estrangulamento.

Muito pode ser aproveitado da experiência adquirida na gerência de bases de dados particionadas que tiram proveito de multiprocessadores, através de bancos de dados distribuídos [Ceri84, Öszu91]. A contribuição do processamento distribuído para banco de dados está no acesso paralelo ao armazenamento das bases de dados e no processamento paralelo desses dados. A dificuldade de sintonizar um sistema de banco de dados paralelo (SBDP) está intimamente ligada ao balanceamento de carga entre os processadores e à minimização do custo de coordenação e sincronização [Laks 90].

O panorama atual dos SBDPs através de experiências práticas ou analíticas tem muito a contribuir para as diversas fontes de paralelismo na gerência de bases de dados. Sendo assim, são apresentadas na Seção II.2, as principais características e técnicas envolvidas no projeto de SBDPs. A Seção II.3 apresenta a descrição e análise de seis projetos/produtos atuais. Finalmente a Seção II.4 apresenta alguns comentários quanto às perspectivas futuras do paralelismo em banco de dados.

A maior parte das Seções deste capítulo consiste de versões atualizadas do material publicado em [Matt91a,b]. Trabalho semelhante a este foi publicado em junho de 1992 [DeWi92] apresentando também uma introdução ao paralelismo em banco de dados e características dos

principais SBDPs. Outro trabalho interessante que apresenta o paralelismo de um modo geral e introdutório, no contexto de sistemas relacionais pode ser encontrado em [Pira90].

II.2 CARACTERIZAÇÃO DE SBDs PARALELOS

As pesquisas na área de paralelismo em sistemas de banco de dados surgem a partir da tecnologia de sistemas de banco de dados distribuídos e das máquinas de banco de dados. Embora essas duas áreas de conhecimento em banco de dados venham sendo exploradas há muito tempo, recentemente elas ganharam um novo impulso com a disponibilidade da tecnologia atual das arquiteturas paralelas, levando aos chamados sistemas de banco de dados paralelos (SBDP). Um levantamento de diversos protótipos/produtos disponíveis é apresentado aqui, através de uma caracterização de SBDPs visando identificar aspectos úteis para a comparação, descrição e o desenvolvimento de SBDPs.

As características dos SBDPs são descritas a seguir através de oito itens considerados representativos, comuns à maioria dos SBDPs.

II.2.1 Acoplamento ao hardware

O acoplamento ao hardware, indica a categoria de concepção do SBDP quanto ao grau de acoplamento ao seu ambiente hospedeiro. Existem basicamente três categorias de concepção de SBDPs descritas a seguir:

a) máquina de bd

Os softwares para as máquinas paralelas de banco de dados são totalmente acoplados ao hardware, ou seja, é projetado um hardware com capacidade de processamento paralelo, específico para suportar as operações de um sistema de banco de dados. Os produtos NonStopSQL [Tand88] e DBC/12 [Alma89] são exemplos representativos dessa categoria. Como protótipos, podem ser citados os sistemas Bubba [Bora90] e Gamma [DeWi90] entre outros.

b) servidor de bd

Os servidores de sistemas de banco de dados paralelos caracterizam-se por seu desacoplamento ao hardware. Em geral, esses servidores não utilizam um hardware específico para banco de dados, mas possuem um sistema operacional que é orientado para banco de dados. Muitas vezes, esse servidor está ligado a várias estações de trabalho que interagem com os usuários realizando um processamento local. Um exemplo típico desta categoria é o projeto do European Data Server (EDS) [Vald90].

c) SBDP geral

Estes sistemas paralelos são gerenciadores de banco de dados desenvolvidos para um computador paralelo de uso geral, ou seja, embora o SBDP rode sobre a arquitetura de uma determinada máquina paralela, esta máquina é de uso geral e não específico para banco de dados. Como exemplo desta categoria está o protótipo XPRS de Stonebraker [Ston88].

II.2.2 Parâmetros da arquitetura hospedeira

A proliferação das tecnologias do processamento paralelo vem apresentando novidades nas áreas de hardware para arquiteturas paralelas, tecnologias de interconexão de processadores e paradigmas de programação. Duncan [Dunc90] apresenta uma definição de arquiteturas baseada em seus componentes e não em máquinas paralelas específicas e utiliza a seguinte **definição de arquitetura paralela:**

> "Uma arquitetura paralela dispõe de um mecanismo de alto nível, explícito para o desenvolvimento de soluções de programação paralela através da utilização de múltiplos processadores (simples ou complexos) que cooperam na solução de problemas por meio de execução concorrente".

Em conjunto com esta definição pode-se utilizar os seguintes conceitos para a caracterização de uma arquitetura paralela:

a) modelo de memória;
b) topologia - mecanismo de comunicação dos processadores;
c) número de nós da configuração;
d) características do processador ou do nó,e,
e) modelo computacional (SIMD, MIMD, etc).

O parâmetro da arquitetura paralela hospedeira que mais afeta o projeto de um SBDP é o modelo de memória que será comentado a seguir.

II.2.2.1 Modelos de memória

A organização da memória (principal e disco) tem influência decisiva no projeto do servidor de dados de um SBDP. Os algoritmos utilizados e principalmente a comunicação para entrada/saída costumam ser orientados para o modelo de memória compartilhada ou para o modelo de memória distribuída (troca de mensagens). Stonebraker [Ston86] batizou esses dois modelos, respectivamente, de 'shared everything' (armazenamento compartilhado), já que tanto a memória principal quanto o armazenamento em disco são compartilhados por todos os

processadores da arquitetura paralela e de 'shared nothing' (armazenamento não compartilhado), onde cada processador possui a sua memória local exclusiva e suas unidades de disco também de acesso exclusivo. Existe, ainda, a denominação 'Shared Disk' [Bhid88b] para um modelo intermediário onde os processadores possuem memória local mas o espaço em disco é compartilhado.

A seguir serão apresentadas algumas características dos dois modelos principais [Ozsu91], [Alma89], [Ston88], comparando-os do ponto de vista do projetista de um SGBD (já que para o usuário final o modelo da memória é transparente).

a) Arquitetura com memória compartilhada ('Shared Everything')

A memória compartilhada é um ponto de estrangulamento em potencial. Se ocorre uma falha no esquema de interconexão da memória com os processadores, o sistema entra em pane. A arquitetura permite poucas extensões para o hardware possuindo um número limitado de processadores, uma vez que são dependentes da largura da banda ('bandwidth'), da tecnologia do barramento ('bus') e da memória. Entretanto, na memória compartilhada, o particionamento dos dados é muito dinâmico pois os dados não estão distribuídos fisicamente permitindo um bom balanceamento de carga, já que num SGBD este fator está intimamente ligado à distribuição dos dados entre os processadores.

b) Arquitetura com memória distribuída ('Shared Nothing')

A memória distribuída, permite expansibilidade para a arquitetura. O software e o hardware são projetados prevendo a extensão do número de processadores. Aumenta a disponibilidade dos dados através do isolamento de falhas e replicação. A programação do projetista do SGBD, de um modo geral, é mais complexa pois existe a necessidade de controle dos dados replicados, e obriga um controle adicional de concorrência para a cooperação de processos. Pode aumentar o desempenho aparente da entrada/saída dos dados através da realização de acessos paralelos às unidades de disco, evitando o uso dos controladores de disco do processador dedicado à entrada/saída. Entretanto, o grau de paralelismo vai depender do particionamento dos dados entre os discos associados aos processadores.

II.2.3 Acoplamento ao sistema operacional

O acoplamento ao sistema operacional (S.O.) da máquina hospedeira do SBDP, analisa quão dependente é o SBDP do S.O. O sistema operacional pode ser voltado para bancos de dados distribuídos, pode possuir características adequadas à gerência de bases de dados, ou ainda, o S.O. pode ser convencional sem nenhuma especificidade para bancos de dados. No último

extremo está o S.O. especificamente desenvolvido para o SBDP. Este é principalmente o caso das máquinas de banco de dados.

II.2.4 Carregamento do código

Esta Seção trata da distribuição das operações de banco de dados entre os processadores, independentemente dos mecanismos que as arquiteturas paralelas possuem para o carregamento de instruções e de dados. Existem sistemas em que o mesmo código (operações) é replicado em todos os processadores, ou seja, todos os processadores podem executar todas as operações. Em outros, existem processadores alocados exclusivamente para determinadas operações. Há ainda o caso em que o carregamento do código é feito, dinamicamente, no momento da execução da operação.

II.2.5 Particionamento dos dados

O particionamento dos dados diz respeito à distribuição dos dados nos meios de armazenamento. O desempenho de uma operação está, diretamente, ligado à fragmentação no armazenamento dos dados de uma base. Os algoritmos levam em consideração essa distribuição no sentido de se obter o melhor balanceamento de carga entre os nós. Diversos trabalhos apresentam contribuições para o balanceamento dos dados [Hira91], [Laks90], [Wolf91], [Kell91]. Entretanto, o balanceamento é ainda mais crítico nas arquiteturas do tipo 'shared nothing', uma vez que a alocação dos dados é em geral física, realizada no momento do carregamento da base. Neste caso, para grandes volumes de dados o ideal é executar-se a operação onde o dado estiver, para evitar grande tráfego na troca de mensagens. Existem, basicamente, dois parâmetros envolvidos no particionamento dos dados:

a) grupamento ('clustering'), e
b) fragmentação horizontal ou vertical.

Existem dois grandes tipos de particionamento de dados, o agrupado ('clustered') e o desagrupado ('declustered') [Ozsu91]. No particionamento **agrupado**, cada relação fica totalmente "contida" em um mesmo nó. Essa técnica pode minimizar o tempo total de execução, se todas as relações envolvidas estiverem no mesmo nó. Entretanto, o particionamento **desagrupado** é o mais utilizado, onde uma relação é fragmentada horizontalmente através dos nós. Neste caso, o tempo de resposta provavelmente será minimizado, porém o tempo total (incluindo inserção e atualização) poderá aumentar devido ao custo de manter a relação fragmentada através de uma determinada função de distribuição.

A fragmentação horizontal pode ser feita de diversas formas e a informação sobre o particionamento da relação é armazenado no catálogo. As principais técnicas de fragmentação horizontal são: circular, faixa de valores e 'hashing'. No caso de uma arquitetura com memória compartilhada ('shared everything') os diversos fragmentos são espalhados em arquivos separados através das unidades de disco. Na arquitetura com memória distribuída ('shared nothing') os fragmentos são distribuídos entre as unidades de disco associadas aos processadores, ou seja, a técnica de particionamento leva em conta o número de processadores com unidades de disco associadas. Outra técnica interessante para a fragmentação horizontal da relação consiste no uso da técnica de faixa de valores para vários atributos da relação ao invés de um único [Ghan92]. Desta forma, são ampliadas as chances do algoritmo de consulta tirar proveito da distribuição. Somente os processadores relevantes são utilizados, deixando os demais livres para outros processamentos em paralelo. Como consequência, essa estratégia diminui o tráfego da comunicação entre os processadores.

Além das técnicas de fragmentação da relação, outras decisões têm que ser tomadas ao longo do particionamento dos dados. Essas decisões envolvem a associação da relação aos processadores [Cope88, Bora88], a utilização de memória cache e a utilização de índices. A associação da relação aos processadores pode ser feita com **desagrupamento total**, onde todos os processadores recebem um pedaço da relação, ou com o **desagrupamento parcial**, onde os fragmentos da relação são distribuídos apenas para um determinado número de processadores, escolhidos de acordo com técnicas de otimização [Cope88, Laks88, Laks90, Yu 91].

II.2.6 Nível de granularidade do paralelismo

O nível de granularidade do paralelismo está ligado à quantidade de operações que será realizada em paralelo no processamento de consultas do SBDP. O nível de granularidade pode ser classificado em três grupos, a saber: entre consultas de diversos usuários, dentro de uma consulta e dentro de operações. Embora não sejam exclusivos, na maioria dos casos, a concepção do SBDP leva em conta o nível de granularidade de paralelismo que deverá ser mais favorecido. Os três níveis são discutidos a seguir.

a) Paralelismo entre consultas

O paralelismo entre consultas, caracteriza-se pela execução de consultas de diversos usuários concorrentemente. O objetivo da implementação paralela é de atender o maior número possível de consultas por segundo de forma concorrente e/ou paralela.

b) Paralelismo dentro de consultas

O objetivo do paralelismo dentro de consultas é de atender o mais rápido possível ao processamento de consultas complexas e ad-hoc. Em geral, aplicações não convencionais (do tipo CAD, IA, Engenharia) caracterizam-se por possuirem poucas, longas e complexas transações

exigindo assim um alto grau de paralelismo dentro da máquina para a própria transação ou consulta complexa. O paralelismo dentro da consulta está ligado à execução de operações de uma mesma consulta em paralelo. Stonebraker [Ston88] propõe para o sistema XPRS a geração de diversos planos de execução da consulta explorando estratégias de paralelismo durante a fase de otimização. São estimados o custo de execução e o plano com custo mínimo é escolhido.

c) Paralelismo dentro de operações:

Quando uma consulta é mapeada para as operações correspondentes que a resolvem, existe uma série de algoritmos incorporados ao SGBD que executam essas operações. Nesse nível de granularidade, procura-se paralelizar o algoritmo responsável pela resolução da operação, explorando ao máximo a disposição (particionamento) dos dados. No caso do modelo relacional, essas operações compõem a álgebra relacional, e o paralelismo dentro das operações da álgebra consiste, em geral, na replicação da operação entre os processadores, e na execução da mesma operação para as partes da relação.

II.2.7 Modelo de dados

A maioria dos protótipos e produtos encontrados na literatura utilizam o modelo relacional na representação dos dados. Entretanto, existem protótipos de SBDPs que estendem o relacional para suportar consultas recursivas, alguma dedução e principalmente objetos complexos. Algumas pesquisas vêm sendo realizadas no sentido de explorar paralelismo em modelos de dados orientados a objetos. Alguns trabalhos abordam a questão dos objetos complexos em modelos de objetos [Hard88], [Khos88], [Mits92] e a distribuição de sistemas de gerência de objetos [Brum88]. Entretanto, o processamento paralelo de objetos ainda está em aberto, uma vez que o particionamento de objetos não é trivial. Essa questão será discutida no capítulo VI.

II.2.8 Algoritmos das operações

Este é um item importante na descrição de um SBDP, mas que não está solidificado o suficiente para uma classificação. Ozsu e Valduriez [Ozsu91] apresentam os algoritmos mais gerais das principais operações da álgebra relacional. Entretanto alguns SBDPs implementam algoritmos específicos, voltados para os parâmetros da arquitetura hospedeira e para o particionamento dos dados. De um modo geral os algoritmos passam por duas fases. A primeira que coordena a distribuição dos dados e a segunda que coordena a execução dos algoritmos seqüenciais localizados em cada processador.

Encontra-se na literatura um vasto material contendo algoritmos paralelos para os diversos operadores da álgebra relacional, sendo [Bitt83] e [Vald84] os pioneiros na introdução

aos algoritmos. Alguns autores visam explorar ao máximo determinadas classes de arquiteturas como em [Rich87] e [Frie90a,b], ou ainda a exploração de modelos de memória onde novos algoritmos são propostos [Chei88], [Schn89], [Murp89], [Wils91].

II.3 SISTEMAS DE BANCO DE DADOS PARALELOS ATUAIS

Foram escolhidos seis SBDPs usando o critério de popularidade na literatura e facilidade de obtenção de informação para apresentação neste trabalho. Esta Seção descreve os seis sistemas de acordo com as características apresentadas na Seção II.2 e apresenta na Tabela II.1 um quadro comparativo destes SBDPs.

EDS [Vald90b, Sala91] - O projeto do European Data Server (EDS) é um projeto Sprit com características ousadas, onde vários grupos de pesquisa estão experimentando diversas alternativas para as técnicas descritas anteriormente. Por ser um projeto em andamento, iniciado em 1988, não existe ainda um protótipo apresentado na literatura e as informações obtidas sobre o projeto foram coletadas em publicações e palestras. O projeto pretende utilizar um sistema operacional voltado para banco de dados. Foi escolhida a linguagem de programação C++ para o desenvolvimento do EDS. Para a utilização/operação do EDS foi realizada uma extensão da SQL - a ESQL [Gard90], para prover capacidade de dedução (através de recursividade) e suporte a objetos complexos com sabor de orientação a objetos (através de TADs e OIDs). O paralelismo dentro de consultas é detectado automaticamente pelo próprio EDS. O nível de granularidade mais explorado é o dentro de consultas e dentro de operações. Uma avaliação da linguagem ESQL está sendo realizada através de um protótipo, o DBS3 que é executado sobre uma arquitetura de memória compartilhada [Berg91] ao contrário da proposta do próprio EDS.

GAMMA [DeWi88, DeWi90] - Gerado na Universidade de Wisconsin, Madison EUA, o projeto de GAMMA teve início em 1984 para gerar uma máquina de banco de dados. Aos poucos o projeto foi mudando, no sentido de se utilizar uma plataforma de hardware mais genérica. GAMMA possui seu S.O. próprio e utiliza o sistema de armazenamento WISS (Wisconsin Storage System). O sistema possui um esquema especial de recuperação na falha de nós. O usuário interage com GAMMA a partir de uma linguagem própria 'ad-hoc' que também funciona embutida em C. O paralelismo dentro de consultas é detectado pelo otimizador. Entretanto, o tipo de fragmentação de tuplas utilizado e o modelo de memória distribuída, não favoreçem muito o paralelismo dentro de consultas e sim dentro de operações. A resolução das operações

da álgebra relacional baseia-se em 'hashing' e é utilizado um algoritmo especial para a junção, desenvolvido pela equipe do Gamma, denominado 'hybrid-join' [Schn89].

BUBBA [Bora90, Cope88] - Gerado no MCC, Austin, Texas EUA, o projeto de BUBBA teve início em 1984 para gerar uma máquina de banco de dados. Aos poucos o projeto foi mudando, no sentido de se utilizar uma plataforma de hardware mais genérica. BUBBA possui uma versão adaptada do UNIX gerando seu S.O. próprio. Está sendo utilizada a linguagem C para o desenvolvimento do SBDP. O usuário interage com BUBBA a partir de uma linguagem própria e autocontida FAD que combina características do modelo relacional estendido com comandos de linguagens de programação. O paralelismo dentro de consultas é detectado automaticamente pelo compilador. O particionamento dos dados em BUBBA é muito especial, utilizando diversos parâmetros como tamanho da relação, freqüência de acesso e tuplas mais concorridas para gerar seu desagrupamento parcial.

XPRS [Ston88, Hong91, Hong92] - Gerado na universidade da Califórnia em Berkley, Califórnia EUA, XPRS (eXtended Postgres on Raid and Sprite) difere dos demais por ser o único com proposta de modelo de memória compartilhada ('shared everything'). Faz uso de um S.O. de uso geral, o Sprite network operating system e a princípio utiliza o computador paralelo SEQUENT Symmetry System. O analisador semântico não detecta fontes de paralelismo dentro de consultas automaticamente, isto sendo feito pelo próprio usuário através do comando 'parallel'. XPRS possui um esquema especial de controle de concorrência e utiliza para o sistema de acesso ao armazenamento secundário, os chamados vetores de disco ('disk array') [Patt88]. Para resolver a operação de junção, o sistema utiliza o algoritmo proposto para o projeto GAMMA denominado 'hybrid-join'. O XPRS conta ainda com um cuidadoso esquema de geração de planos de execução pelo otimizador, levando em conta diversos aspectos do paralelismo como utilização de índices e alocação de processadores.

DBC [Alma89, Tera84] - DBC/1012 é a máquina de banco de dados da Teradata Corporation que visa atender aplicações que privilegiam o paralelismo dentro da consulta. Para resolver a operação de junção, o sistema utiliza algoritmos convencionais. Sendo uma máquina de banco de dados, ela possui seu próprio sistema operacional distribuído, voltado para banco de dados. Funciona como um computador 'back-end' baseado em SQL. Possui suporte para diversos equipamentos hospedeiros tipo MVS, VM/CMS, e os compatíveis com UNIX ou DOS.

NonStopSQL [Tand88, Alma89] - É a máquina de banco de dados da Tandem Computers que visa atender a classes de aplicações que necessitam paralelizar ao máximo o atendimento de vários usuários concorrentemente, ou seja, o paralelismo entre consultas. Este produto vem sendo muito bem aceito em aplicações bancárias. Para resolver a operação de junção, o sistema utiliza algoritmos convencionais

baseados em 'hashing'. Sendo uma máquina de banco de dados, ela possui seu sistema operacional específico, voltado para bancos de dados distribuídos. É um computador baseado em SQL que possui ênfase em tolerância a falhas com duplicação de todas as conexões críticas.

A seguir apresenta-se um quadro comparativo com os diversos sistemas descritos anteriormente. A classificação das características dos SBDPs levou em conta aspectos do harware, através da arquitetura hospedeira, e do software, através do particionamento dos dados e do modelo de dados adotado. Os itens utilizados nas colunas da Tabela II.1 correspondem às características dos SBDPs apresentadas na Seção II.2. Os espaços preenchidos com 'Não Def' correspondem à indefinição do item no projeto e a '?' corresponde à falta de informação na literatura disponível.

	ARQUITETURA HOSPEDEIRA				PARTICIONAMENTO DADOS			MODEL
	Modelo Memória	**Topolog**	**N. Nós**	**Process**	**Fragm**	**Grupam**	**Distr**	
EDS	Distr	?	2-252	Intel	Horiz	Desagr parcial variav	Faixa	Rel Ext
GAMMA	Distr	HCubo	32	Intel iPSC/2	Horiz	Desagr total	Circul Faixa Hash	Rel
BUBBA	Distr	HCubo	40	Motrla 68020	Horiz	Desagr parcial	Faixa Hash	Rel Ext
XPRS	Compar	Não Def	Não Def	Não Def	Horiz	Desagr parcial	Faixa Hash	Rel Ext
DBC	Distr	Arvore	2-1024	Intel 80386	Horiz	Desagr total	Hash	Rel
Non Stop SQL	Distr	Bus	2-32	?	Horiz	Desagr total	Faixa	Rel

Tabela II.1 - Classificação de SBDPs

Observando-se a Tabela II.1, verifica-se uma tendência no desacoplamento do SBDP ao hardware. Diversos projetos que, inicialmente, possuíam seu hardware próprio, partem agora para a utilização de máquinas genéricas e eventualmente com sistema operacional próprio ou dotado de características voltadas para banco de dados.

Em relação aos parâmetros da arquitetura hospedeira, nota-se que embora a topologia possa influenciar o projeto de alguns algoritmos de operações do SBDP, é o modelo de memória que mais afeta o projeto de um SBDP. Não existe um consenso em relação ao modelo mais eficiente. Alguns trabalhos analíticos com simulações, [Bhid88a,b] e [Laks89 e 90], apontam a arquitetura com memória compartilhada como a solução que atinge o maior desempenho, em análises que envolvem até 32 processadores. Entretanto, vários projetos como Gamma, Bubba, EDS, entre outros apostam na expansibilidade da arquitetura com memória distribuída, o que já vem ocorrendo tanto ao nível de hardware quanto software. Em trabalho recente, DeWitt e Gray [DeWi92] fazem uma análise qualitativa das três arquiteturas e apostam fortemente no sucesso do crescimento do modelo 'shared nothing'.

O particionamento dos dados é o item onde os sistemas concentram a maior parte das pesquisas aproveitando a similaridade com a fragmentação de bancos de dados distribuídos. Especificamente novas técnicas de desagrupamento parcial vêm sendo estudadas, pois embora o desagrupamento total seja mais facilmente implementado e favoreça consultas que podem ser limitadas a um determinado processador, o desagrupamento total tem suas desvantagens para alguns casos de junção e principalmente no espalhamento de relações com cardinalidade baixa. A proposta do sistema Bubba [Cope88] que leva em conta padrões de acesso para distribuir os dados parece ser a mais indicada. Apesar da importância do sistema Grace[Fush86] não foi possível obter material suficiente para sua análise.

II.4 COMENTÁRIOS FINAIS

Foram levantadas na Seção II.2 diversas fontes de paralelismo na gerência de bases de dados. Observa-se que aplicações não convencionais podem se beneficiar do ganho em desempenho proporcionado pelas arquiteturas paralelas na execução de consultas complexas. O paralelismo em sistemas de gerência de objetos ainda é um tema em aberto, entretanto estão sendo realizadas pesquisas onde o paralelismo é utilizado no auxílio à busca de métodos na hierarquia de classes e na manipulação de coleções de objetos. O capítulo VI analisa em detalhes a questão do paralelismo nas consultas "complexas" e no contexto da orientação a objetos, propondo uma solução onde convivem o agrupamento de objetos (adotado na maioria dos SGBDOOs) e o desagrupamento do processamento paralelo.

Capítulo III

O SERVIDOR PARALELO DE CONSULTAS RELACIONAIS

III.1 PRELIMINARES

No sentido de ganhar experiência na distribuição e sincronização de operações da álgebra relacional, foi implementado na máquina paralela desenvolvida na COPPE, o NCP I [Amor91], um protótipo de SBDP, o PARBASE, utilizando o ambiente de programação paralela STRAND [Fost90]. Realizaram-se várias medidas de desempenho no PARBASE onde constatou-se o ganho no desempenho sempre que o número de processadores envolvidos no processamento da consulta aumentava, além de ser observada a estabilidade do sistema em relação à variação do número de tuplas nas relações envolvidas [Matt91a]. Os resultados obtidos foram bastante animadores uma vez que atingiram os ganhos considerados ideais.

Neste capítulo, é apresentado na Seção III.2 o ambiente de desenvolvimento do protótipo PARBASE. As características, os componentes e a implementação do PARBASE são descritos na Seção III.3. Comentários sobre o desempenho e comportamento do PARBASE são realizados na Seção III.4 com o auxílio de gráficos de tempo de resposta e de ganho em desempenho. A maior parte do material das Seções III. 2, 3 e 4 foi extraída de [Matt91]. Finalmente a Seção III.6 sumariza as principais conclusões obtidas à partir da implementação do PARBASE.

III.2 O AMBIENTE DE DESENVOLVIMENTO DO PARBASE

O ambiente de desenvolvimento do PARBASE pode ser dividido em duas partes. A primeira é o ambiente computacional caracterizado pela máquina paralela da COPPE/UFRJ, o NCP I. A segunda consiste no ambiente de programação englobando o sistema operacional paralelo Helios [Heli88] e o ambiente de programação paralela STRAND [Fost90], [Arti90]. O sistema operacional Helios é uma versão paralela do Unix. A vantagem da utilização do Helios está na sua portabilidade, pois sendo genérico e rodando em diversas máquinas, facilita a integração do SBDP com outros sistemas. STRAND é uma linguagem de programação com características avançadas tanto para a programação seqüencial convencional quanto para a programação paralela.

III.2.1 Ambiente computacional

O computador paralelo NCP I [Amor91] é um protótipo pré-industrial projetado e implementado na COPPE/UFRJ. O NCP I é baseado numa arquitetura de memória distribuída com topologia básica hipercúbica. A comunicação entre os nós de processamento é feita através dos elos seriais de comunicação a uma taxa de 20 Mbits/s. A arquitetura de um nó é mostrada na Figura III.1. Como pode ser visto, cada nó possui um transputer T800 (32 bits, 25 MHz com 4 Mbytes de memória e um microprocessador i860 (64 bits, 40 MHz) com 8 Mbytes de memória, esta sendo compartilhada com o transputer. Além disso, existem 16 Kbytes de memória vetorial para troca rápida de informação entre os dois processadores.

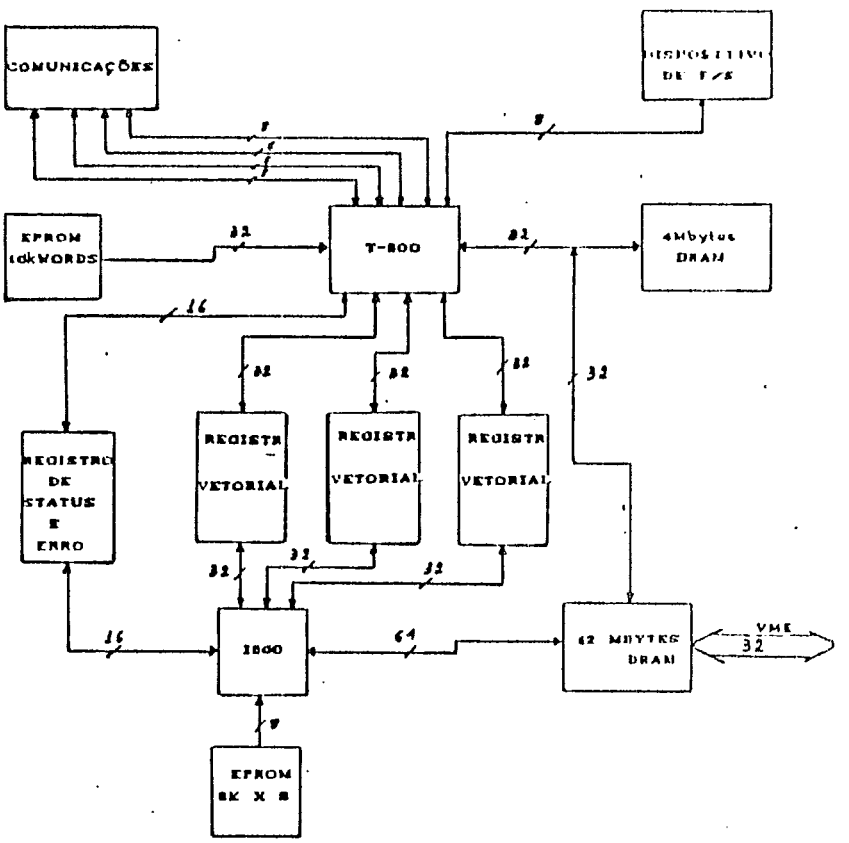

Figura III.1 - Arquitetura do nó do NCP I

A comunicação externa é feita através de um 'crossbar switch' e de um barramento VME. O primeiro permite a interligação do nó com até 2048 nós, utilizando comunicação serial. Este 'crossbar switch' permite desacoplar a arquitetura da topologia hipercúbica quando necessário. A interface VME fornece uma comunicação paralela à base de dados podendo ser utilizada também para comunicação entre nós vizinhos. Neste barramento pode ser colocado, além de controladores de disco, interface para rede e

interface para outros computadores hospedeiros. A capacidade do protótipo é de 1,28 GFlops, 800 Mips e 192 Mbytes de memória.

As especificações e características do NCP I estão sendo experimentadas e exploradas no nível do hardware. A versão do NCP I disponível para os usuários e utilizada neste trabalho, é o modelo T8 do NCP I e corresponde a 8 nós interligados nessa topologia hipercúbica. Cada nó contém um transputer e 2 Mbytes de memória. O computador hospedeiro é um IBM PC-AT. A capacidade do T8 é de 16 MFlops, 80 Mips e 16 Mbytes de memória.

III.2.2 Ambiente de programação STRAND

A linguagem STRAND possui diversas características peculiares que facilitam o desenvolvimento de algoritmos paralelos através de seu alto nível de programação descritos a seguir.

i) STRAND como linguagem de programação convencional

STRAND é uma linguagem declarativa e que impõe recursividade em sua programação. Possui tipos de dados estruturados como listas, tuplas e conjuntos, o que facilitou o desenvolvimento do PARBASE. Entretanto, todas as suas variáveis são univaloradas, o que acrescenta alguma dificuldade de programação. STRAND é interpretada e foi projetada para a programação multi-linguagens, ou seja, possui mecanismos de integração com Fortran e C para o desenvolvimento de código local seqüencial e reaproveitamento de código. A linguagem apresenta características de Prolog e Smalltalk, mas não chega a implementar os paradigmas da programação em lógica ou da orientação a objetos.

ii) STRAND como linguagem de programação paralela

Um programa STRAND caracteriza-se por um pool de processos comunicando-se através de variáveis que são argumentos dos processos. Os processos são obtidos a partir da execução de procedimentos que são escritos como regras da forma:

r(p1,..,pn) :- <guarda de cláusulas> | <corpo>.

onde:

r identifica uma regra de n parâmetros,

" **:-** " é o símbolo de implica em,

<guarda de cláusulas> são as cláusulas a serem verificadas para a execução do corpo,

" | " delimita as cláusulas do corpo, e

<corpo> é um conjunto de processos.

O exemplo de um procedimento de nome teste pode ser definido por:

```
teste(Numero, Resposta) :- Numero > 0 | Resposta := positivo.
teste(0, Resposta) :- Resposta := zero.
teste(Numero, Resposta) :- Numero < 0 | Resposta := negativo.
teste(Numero, Resposta) :- otherwise | Resposta := erro.
```

STRAND é altamente concorrente, aproveitando a disponibilidade dos processadores para executar (reduzir) os processos do pool. Não existem comandos do tipo 'parallel do', 'fork' ou 'join'. Em princípio todos os processos podem ser executados concorrentemente e em paralelo, o que implica num sincronismo que é controlado pelas variáveis (os argumentos dos processos). Um processo que depende da instanciação de uma determinada variável, fica aguardando no pool até que ela receba um valor.

Outra característica interessante é a sua associação direta entre código e processadores. No caso de um SBDP, o código da seleção pode ser associado a **n** processadores diretamente para que a seleção seja executada em paralelo.

III.3 O PROTÓTIPO PARBASE

PARBASE é um protótipo de sistema de gerência de bases de dados paralelo de uso geral que implementa alguns operadores da álgebra relacional e é independente da máquina paralela hospedeira. Na implementação não houve preocupação com o desempenho em comparação com outros sistemas. Buscou-se a construção de um sistema que provê a funcionalidade mínima de um SGBD e que serve como uma plataforma onde diversas técnicas de paralelismo podem ser experimentadas, contribuindo para o desenvolvimento de um SBDP que atenda a necessidades reais.

III.3.1 Características do protótipo PARBASE

O **nível de granularidade** do processamento paralelo de consultas mais explorado no PARBASE é o paralelismo dentro de operações. Entretanto, uma consulta no PARBASE não necessariamente terá suas operações executadas sequencialmente. É possível haver paralelismo entre as operações de uma consulta controlado pelo PARBASE. Por ser mono-usuário, o PARBASE não provê o paralelismo entre consultas de vários usuários.

Apesar do **modelo de memória** da máquina NCP I ser do tipo 'shared nothing' (armazenamento distribuído), a versão do NCP I utilizada para o desenvolvimento e execução do PARBASE é do tipo 'shared disk', ou seja, cada processador possui sua memória local e todos os processadores compartilham a mesma unidade de disco. Entretanto, como o modelo da memória da máquina deverá ser de armazenamento distribuído, optou-se por um particionamento de dados mais adequado à memória distribuída.

A **fragmentação das relações** é horizontal e optou-se por um desagrupamento total para as relações, ou seja, se a máquina possui oito processadores, a relação é fragmentada em oito partes.

A **política de distribuição** de tuplas de uma relação entre os diversos fragmentos é a de faixa de valores. No momento da criação de uma relação o usuário deve especificar o atributo de distribuição e as respectivas faixas de valores. Esta estratégia de distribuição, assim como o 'hashing', tira proveito da localidade conhecida para as faixas de valores e com isso atinge um ganho no desempenho. Num SBDP completo, estas faixas poderiam ser obtidas automaticamente pelo otimizador, através de amostragens ou monitorações.

III.3.2 A implementação do PARBASE

Os componentes do PARBASE estão representados na Figura III.2. Essa implementação aproveita algumas sugestões fornecidas por Steer [Stee89] para o desenvolvimento de um SGBD concorrente através da linguagem Strand.

O usuário interage com o PARBASE através de comandos associados aos operadores da álgebra relacional, com a seguinte sintaxe:

```
select (Id-us, Atributos, Relação, Condição, Resultado)
join (Id-us, Relação1, Relação2, Condição, Resultado)
insert (Id-us, Relação, Tupla, Resultado)
delete (Id-us, Relação, Condição, Resultado)
update (Id-us, Relação, Condição, Atributos, Resultado)
create (Id-us, Relação, Chave, Especificação, Resultado)
drop (Id-us, Relação, Resultado)
```

No sentido de facilitar o desenvolvimento do protótipo, não foi implementado um processador para analisar a linguagem de consulta. Assim a sintaxe elaborada simplifica as verificações léxicas e sintáticas no PARBASE. O usuário pode submeter um ou mais

comandos através de uma lista ao PARBASE. O sistema se encarregará de prover o máximo de paralelismo entre os comandos da lista do usuário.

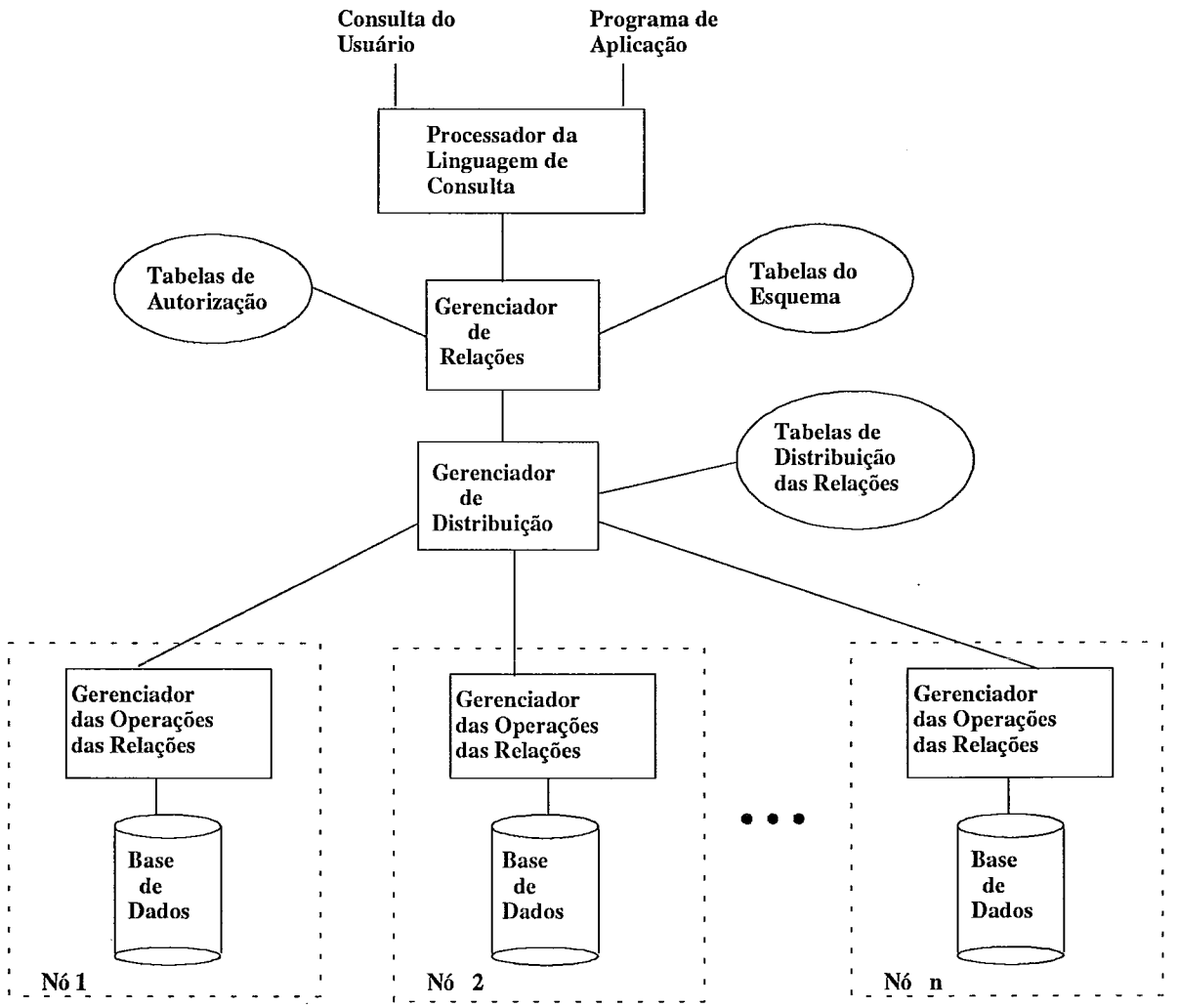

Figura III.2 - Componentes do PARBASE

O **Gerenciador de Relações**, é responsável por algumas verificações semânticas do tipo existência da relação envolvida, nível de autorização do usuário para determinadas operações, etc.

O **Gerenciador de Distribuição** é o responsável pela análise dos atributos envolvidos na operação e pela conseqüente decisão sobre quais processadores estarão envolvidos na execução da operação. O Gerenciador de Distribuição consulta as tabelas de distribuição e, eventualmente, impõe alguma sincronização na execução das sub-

operações. Na identificação dos processadores e fragmentos para operações que envolvem condições, o Gerenciador de Distribuição verifica se somente um processador será ativado ou se todos executarão a operação, cada um com seu fragmento da relação. Ao término da operação, esse mesmo Gerente se encarrega de fazer a união da relação resultado. Quando somente um processador é envolvido na operação, os outros processadores executam as operações restantes da consulta do usuário.

No caso do operador de junção, o Gerente de Distribuição identifica se a junção pode ser executada totalmente distribuída ou se a junção será executada parcialmente distribuída. A execução da junção totalmente distribuída ocorre quando as duas relações envolvidas estão fragmentadas sobre seus respectivos atributos da condição de junção. Neste caso, as operações de junção são executadas em paralelo cada uma em um processador e é realizada a união dos resultados parciais no final. A execução da junção parcialmente distribuída ocorre em duas fases. Inicialmente o Gerenciador de Distribuição escolhe a relação de menor cardinalidade, junta todos os seus fragmentos para que em seguida a envie completa a todos os processadores para que cada um execute a junção com a parcela da outra relação operando, associada ao seu nó.

III.4 ANÁLISE DO DESEMPENHO DO PARBASE

São apresentados aqui os resultados de uma avaliação do desempenho do protótipo do SBDP PARBASE na máquina paralela NCP I. Um sistema paralelo ideal apresenta duas propriedades chave [DeWitt 90]: a aceleração linear e a expansibilidade linear. A aceleração linear é medida através do ganho no desempenho considerando-se um número fixo de tuplas da relação à proporção que aumenta-se o número de processadores. O ganho pode ser definido então pela razão: tempo de processamento com 1 processador sobre tempo de processamento sobre N processadores. Já a expansibilidade linear é obtida quando aumenta-se a carga do SBDP e o tempo de processamento aumenta proporcionalmente. Além disso, espera-se que o crescimento do hardware corresponda a um decréscimo proporcional no tempo de processamento [Nuss91].

Foi realizada uma série de testes com os operadores de seleção e junção onde foram analisados os tempos de resposta medidos para variações no número de tuplas das relações e no número de processadores utilizados na configuração da arquitetura. Os testes foram aplicados no sentido de medir-se a aceleração e a expansibilidade do PARBASE.

As relações utilizadas para o 'benchmark' são instâncias de uma base de dados real. Cada relação possui quatro atributos inteiros de 4-bytes e quatro atributos do tipo

cadeia de caracteres totalizando 97 caracteres. Foram realizados testes sobre relações de 100, 1000 e 3000 tuplas. Cada relação possui dois atributos sem duplicatas e não estão relacionados entre si. Um desses atributos de valor único, foi utilizado para a chave de distribuição em todos os casos.

Todos os testes foram repetidos para configurações de 1, 4 e 8 processadores no NCP I. Os tempos foram medidos a partir do momento que o usuário submeteu a consulta ao PARBASE até a finalização da consulta. Foram incluídas neste tempo todas as passagens da consulta pelos diversos gerentes do PARBASE.

III.4.1 Seleção

O desempenho do PARBASE foi explorado para vários tipos de seleção sobre relações cujo tamanho também variava, baseado no 'benchmark' do SBDP Gamma [DeWi88]. O objetivo dessas medidas foi o de analisar o comportamento do PARBASE à medida que o tamanho das relações aumentava. Em condições ideais, espera-se que o tempo de resposta aumente de acordo com uma função linear sobre o tamanho das relações operando, dado que as configurações da máquina são constantes.

Foram realizados testes sobre quatro tipos de consulta envolvendo a seleção. Em três consultas variou-se o fator de seletividade com 0%, 1% e 10% para as relações resultado. Na quarta consulta o resultado da seleção envolvia somente uma tupla.

As Tabelas III.1 e III.2 mostram os resultados obtidos para os diversos tipos de seleção sobre relações de tamanho 100, 1000 e 3000 tuplas para as configurações de 4 e 8 processadores ligados em hipercubo. Como as relações eram de uma base real foram realizadas pequenas adaptações para se atingir exatamente os fatores de seletividade.

	Número de tuplas da relação fonte		
Consulta	100	1000	3000
seletividade 0%	5,40	50,26	175,87
seletividade 1%	5,42	50,94	174,59
seletividade 10%	5,44	47,85	173,15
seleção 1 tupla	4,74	44,14	128,76

Tabela III.1 - Consultas de seleção para 4 processadores
(todos os tempos em segundos)

	Número de tuplas da relação fonte		
Consulta	100	1000	3000
seletividade 0%	4,45	24,26	76,23
seletividade 1%	4,40	23,27	74,25
seletividade 10%	4,45	23,29	76,23
seleção 1 tupla	3,34	18,69	61,29

Tabela III.2 - Consultas de seleção para 8 processadores
(todos os tempos em segundos)

Algumas conclusões podem ser obtidas das Tabelas III.1 e III.2. Em primeiro lugar, conforme esperado, observa-se que o tempo de execução de cada consulta aumenta quase que proporcionalmente com o tamanho das relações operando para cada número de processadores testados, ou seja, 1, 4 e 8. A única exceção ocorre para as seleções da relação de 100 tuplas quando aumentam para 1000 tuplas na configuração de 8 processadores. Nas configurações de 1 e 4, o tempo de resposta cresce em função do aumento de tuplas nas relações. Essa exceção é explicável pela má distribuição de carga que ocorreu entre a relação de cardinalidade 100 e a configuração de 8 processadores.

Foi observado que não existe muito ganho em desempenho ao se utilizar muitos processadores para uma relação com poucas tuplas. Em casos assim, o preço que se paga para manter os diversos fragmentos e a comunicação entre os processadores não se traduz em custo/benefício. Esta é uma situação em que o desagrupamento total não funciona bem.

Como uma segunda conclusão pode ser observado o ganho em desempenho sempre que se aumenta o número de processadores e consequentemente o número de fragmentos que compõem a relação.

Cabe ainda observar que os melhores tempos de resposta estão sempre para a seleção de uma tupla, já que esta é a única consulta em que o Gerente de Distribuição do PARBASE tira proveito da distribuição da relação e não necessita percorrer toda a relação, mas somente o fragmento envolvido.

A Figura III.3 apresenta o tempo médio de resposta para consultas com fator de seletividade 0%, 1%, 10% e seleção de 1 tupla, para a relação de 1000 tuplas em função do número de processadores utilizados. A Figura III.4 apresenta a curva de ganho relativo ao tempo de processamento correspondente à Figura III.3. Pode ser observado que o ganho é quase linear para os fatores de seletividade.

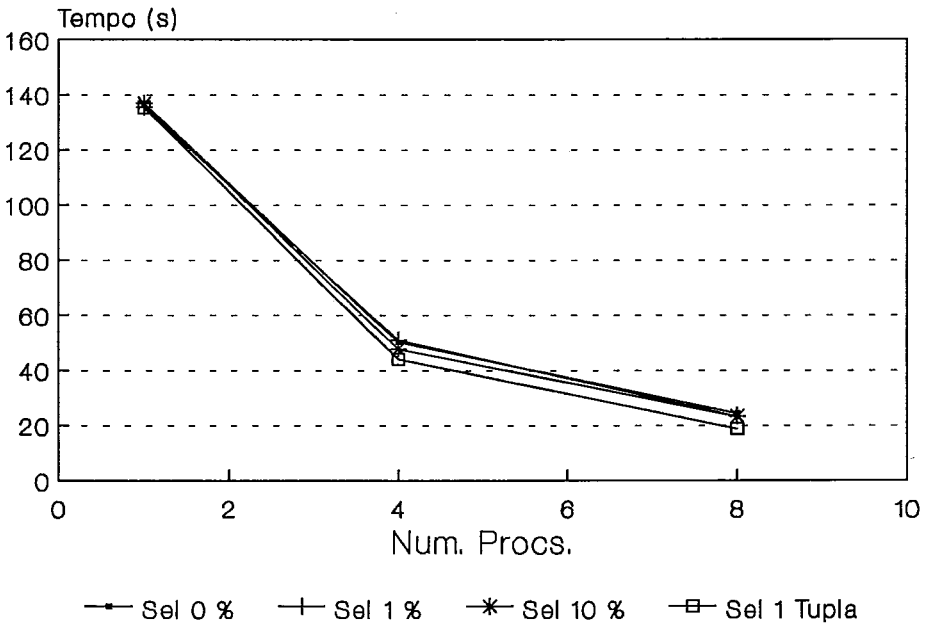

Figura III.3 - Tempos de resposta para a tabela de 1000 tuplas

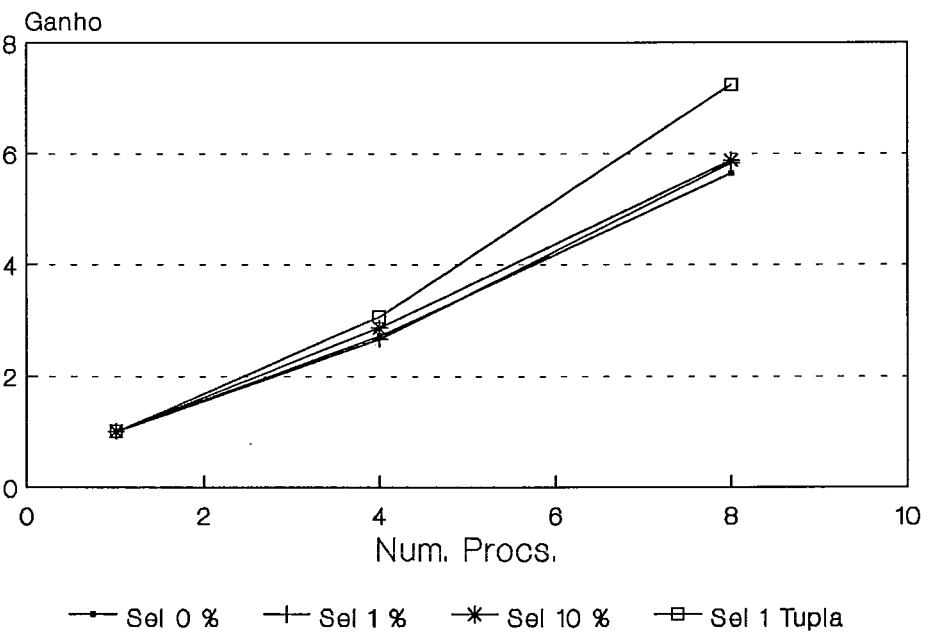

Figura III.4 - Ganhos obtidos para a tabela de 1000 tuplas

III.4.2 Junção

Foram realizados diversos testes sobre o operador de junção do PARBASE e são apresentados aqui os resultados para junções ocorrendo entre tabelas de 100x1000 tuplas. O objetivo das medidas das consultas envolvendo o operador de junção foi o de analisar o desempenho do PARBASE à proporção que o número de processadores e fragmentos da relação aumentava.

Dois tipos de junção foram testados. No primeiro tipo, a junção opera sobre os atributos de distribuição das duas relações operando. Neste caso o PARBASE executa a operação de junção totalmente distribuída. No segundo tipo de teste, a junção opera sobre atributos não chave de distribuição das relações operando. Neste caso, o PARBASE executa a operação de junção em duas fases, sendo que a primeira é para a redistribuição da relação de menor cardinalidade.

Conforme era esperado, pode ser observada a melhoria do desempenho através das curvas de tempo de resposta (Figura III.5) e do ganho relativo. Estas curvas também mostram um tempo de resposta mais rápido para a junção totalmente distribuída. Este ganho só não é mais significativo devido ao fato da relação de menor cardinalidade conter apenas 100 tuplas.

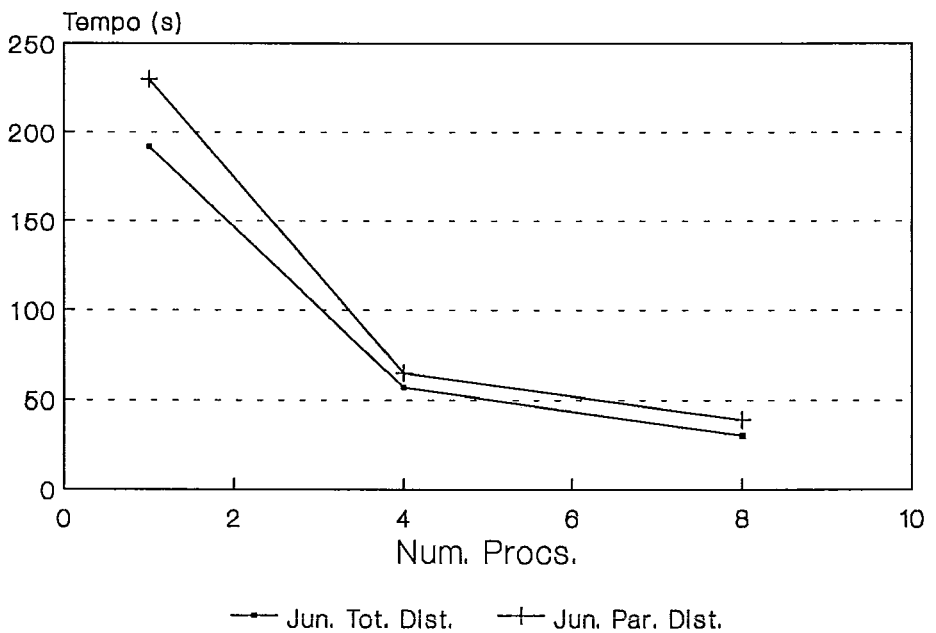

Figura III.5 - Tempos de resposta para a junção de 100 x 1000 tuplas

III.5 COMENTÁRIOS SOBRE A EXPERIÊNCIA REALIZADA

A proposta do PARBASE difere basicamente dos sistemas revistos no capítulo II por dois fatores. O primeiro é a sua independência da máquina paralela hospedeira e do sistema operacional. O segundo fator é a obtenção das medidas de desempenho na plataforma real onde se deseja desenvolver o SBDP produto. Optou-se pelo desenvolvimento de um protótipo simples com as características básicas de paralelismo, sobre o qual várias técnicas alternativas pudessem ser avaliadas e exploradas. Os gerenciadores do PARBASE dão tratamento uniforme para todas as operações implementadas facilitando a modificação e extensão do sistema.

Foram obtidas diversas medidas de desempenho e reportadas aqui algumas aferições para as operações de seleção e junção, já que representam respectivamente operações totalmente distribuídas e diretamente paralelizáveis e operações parcialmente distribuídas. Os resultados obtidos foram de acordo com o esperado, ou seja, verificou-se um crescimento uniforme no sistema a partir do tamanho das relações e um ganho no desempenho quase sempre linear em relação ao acréscimo dos processadores nos casos analisados.

Cabe observar que, o fato de se possuir um protótipo que não foi desenvolvido especificamente para um determinado sistema operacional ou máquina paralela, permite que:

i) novas medidas de desempenho possam ser realizadas e comparadas em outras máquinas (como por exemplo a máquina da Intel i860),

ii) haja mais independência dos dados, e

iii) a construção e migração de aplicações para o SBDP sejam facilitadas.

Finalmente, a experiência obtida facilitou a visualização de fontes de paralelismo para SBDPs com dados orientados a objetos. Conforme apresentado no capítulo VI diversos gerentes do PARBASE puderam ser aproveitados no protótipo que contempla a orientação a objetos.

Capítulo IV

A ARQUITETURA DO SGO GEOTABA

Para explorar paralelismo no contexto da orientação a objetos, foi necessária a especificação de uma arquitetura de sistema de gerência de objetos que comportasse o uso de paralelismo. Esta arquitetura foi especificada no contexto do GEOTABA - um sistema de gerência de bases de dados orientado a objetos (SGBDOO), adequado a ambientes de desenvolvimento de software. Foram estudadas e desenvolvidas técnicas de orientação a objetos para gerência de objetos e processamento de consultas em arquiteturas de computadores seqüenciais. Essas técnicas foram avaliadas e exploradas através da implementação do GOA - o Gerente de Objetos Armazenados do GEOTABA com capacidade de avaliação de predicados de consulta. A implementação do GOA forneceu subsídios para o desenvolvimento do PARGOA - o Gerente Paralelo de Objetos Armazenados do GEOTABA, para ser utilizado em máquinas paralelas.

IV.1 PRELIMINARES

Como nas linguagens de programação orientadas a objetos (LPOOs) [Gold83] [Stro86] e nos bancos de dados orientados a objetos [Banc88], [Kim 90b], [Fish89], [Maie86], o GEOTABA define objetos como unidades de informação que associam dados a seu comportamento próprio (funções específicas de cada tipo de dado). O modelo de objetos do GEOTABA [Mont93], apesar de incorporar conceitos próprios aos SGBDOOs e LPOOs, como objetos, classes, herança e polimorfismo, representa relacionamentos e restrições de integridade entre objetos, de um modo diferente destes sistemas, numa abordagem mais próxima aos modelos semânticos de dados [Chen76].

Para dar suporte ao modelo de objetos do GEOTABA, foi projetada uma arquitetura de SGBDOO com características de extensibilidade. A preocupação com o suporte à aplicações não convencionais e consequentemente com o desempenho do SGBDOO levou à especificação de uma **plataforma do tipo cliente/servidor com exploração de paralelismo no servidor de objetos.**

Para o desenvolvimento dos módulos da arquitetura do GEOTABA várias opções foram analisadas no sentido de se explorar paralelismo na gerência dos objetos. Inicialmente pensou-se

na implementação do SGBDOO utilizando-se a linguagem POOL2 [Amer91, Huls91, Aalb91], uma linguagem de programação orientada a objetos paralela, ou seja, a programação adota o modelo do paradigma de objetos e o código gerado é executado sobre uma máquina paralela. Entretanto, a utilização da POOL2 foi abandonada devido às limitações do modelo de objetos da linguagem e por gerar código específico para uma máquina de difícil acesso, o que dificultaria sua integração com a arquitetura mais abrangente do Projeto TABA.

Considerando que as maiores fontes de paralelismo estão nas operações de manipulação de objetos, incluindo consultas, decidiu-se limitar o escopo do paralelismo ao módulo executor da linguagem de manipulação de objetos. Neste contexto, foi tentada a utilização de servidores de objetos de protótipos já prontos como por exemplo os sistemas O_2 [O2Te91], XOS [Gibb90], SOS [Schi92, Uhl 91] e GEODE [Bell90]. A idéia da utilização desses protótipos era realizar a exploração do paralelismo no contexto de uma rede de estações de trabalho e, no caso do acesso ao código fonte do sistema, a exploração de algoritmos específicos de paralelismo. Entretanto, foram encontradas diversas dificuldades, dentre elas a pouca flexibilidade no esquema de armazenamento dos objetos, além do problema de se trabalhar com um código "importado". Desta forma, além da especificação de uma arquitetura de SGBDOO que contemplasse paralelismo, foi necessária a especificação e implementação de um gerente de objetos sequencial para em seguida explorar suas fontes de paralelismo sobre um hardware paralelo.

Sendo assim, este capítulo apresenta, inicialmente, algumas características gerais sobre tecnologias de arquiteturas de SGBDOOs para em seguida descrever a arquitetura do GEOTABA. A experiência de implementação do servidor de objetos seqüencial é descrita no próximo capítulo.

IV.2 ARQUITETURAS PARA SGBDOOs

Com o crescente interesse nas aplicações de banco de dados ditas não convencionais, vários são os rumos que a pesquisa de novas tecnologias para essas aplicações têm tomado. Surgiram inicialmente, projetos de SGBDs específicos para uma determinada aplicação e propostas de extensão dos SGBDs já existentes, tipicamente aqueles baseados no modelo relacional. Particularmente, na COPPE, foram realizadas experiências com o SGBD baseado no modelo relacional COPPEREL [Matt87] no sentido de capacitá-lo à gerenciar objetos complexos e permitir o armazenamento de dados de tamanho arbitrário (campo longo), entre outros [Trot89].

O reaparecimento das LPOOs deu novo impulso aos modelos de dados orientados a objetos. Esses modelos se tornaram a base de construção de SGBDs não convencionais baseados na orientação a objetos. Segundo Dittrich [Ditt86], em artigo precursor na definição de

SGBDOOs, o uso da orientação a objetos na modelagem de dados de um SGBD surge para diminuir o 'gap' semântico extistente entre a modelagem semântica de uma aplicação e seu mapeamento para o modelo relacional dos SGBDs.

A falta de um modelo de dados com um embasamento teórico e alguma confusão de termos levou um grupo de pesquisadores adeptos da orientação a objetos em bancos de dados à publicação de um manifesto da orientação a objetos [Atki90]. O manifesto da orientação a objetos tenta padronizar os conceitos básicos de um SGBDOO e apresenta a tecnologia da orientação a objetos como a solução para os problemas das aplicações não convencionais. Esse manifesto gerou a publicação de um outro trabalho [Ston90] contestanto-o, tentando mostrar que nem tudo seria resolvido com a orientação a objetos e que alguns pontos importantes para a gerência de dados permaneciam em aberto na solução SGBDOO. Neste trabalho, Stonebraker apresenta, então, os chamados SGBDs de terceira geração que se baseiam em extensões do modelo relacional de dados.

Paralelamente a essas discussões, os primeiros protótipos de SGBDOOs desenvolvidos O_2, Orion, GemStone, Vbase, Exodus, entre outros começaram a fornecer subsídios para críticas, aperfeiçoamentos e extensões dentro da tecnologia adquirida e descrita em vários artigos da literatura. Diversas experiências puderam ser realizadas com os protótipos (por exemplo, Damokles, O_2) distribuídos para universidades a título de testes e validações.

A partir de 1990 surgiram livros que muito contribuiram para a tecnologia da construção de SGBDOOs. O livro sobre o O_2 [Banc92] descreve e discute as soluções, para os diversos componentes de um SGBDOO, adotadas na contrução do O_2. O livro de Kim [Kim 90a] embora não seja específico sobre a construção do Orion (denominado Itasca em sua versão comercial) apresenta como soluções tecnológicas para SGBDOOs, aquelas escolhidas para o projeto do Orion (Itasca). O trabalho de Kim apresenta ainda um levantamento e análise dos principais protótipos/produtos de SGBDOOs existentes.

O livro de Cattell [Catt91] tenta ser o mais genérico possível, descrevendo aspectos funcionais e de arquiteturas de um SGBDOO e utiliza uma vasta gama de exemplos e descrição de protótipos atuais. A apresentação da arquitetura do GEOTABA utiliza a descrição abstrata de SGBDOOs proposta por Cattell. Entretanto, Cattell salienta que são muitos os temas ainda em pesquisa dentro da construção de SGBDOOs, como por exemplo, a avaliação de consultas.

Dentro das perspectivas futuras de SGBDOOs, Cattel aponta a necessidade de exploração de paralelismo, sem no entanto, apresentar contribuições nesse sentido. Outros trabalhos [DeWi90, Öszu91] também apostam no sucesso das máquinas paralelas para SGBDOOs, porém não foram encontradas na literatura, soluções de paralelismo para a gerência de objetos. Alguns trabalhos apresentam propostas de SGBDs com modelos de dados com orientação a objetos, mas que, no entanto, mapeiam esses dados para o modelo relacional no nível

interno de manipulação. Nestes casos, a solução de paralelismo é aplicada sobre o modelo relacional de representação de dados.

IV.2.1 Plataformas Cliente/Servidor em arquiteturas de SGBDOOs

Com a evolução na tecnologia de comunicação entre sistemas, foi ganhando impulso a técnica de dividir os componentes de um sistema entre processos cliente e servidor. A idéia é aumentar o desempenho do sistema aproveitando a capacidade de processamento do servidor e do cliente concorrentemente ou em paralelo.

Em trabalho recente, Sinha [Sinh92] comenta sobre os diversos aspectos tecnológicos do paradigma cliente/servidor e apresenta um modelo computacional para este paradigma apontando as seguintes vantagens para o modelo cliente/servidor:

a) O cliente aproveita a capacidade gráfica de sua estação para o processamento da interface com o usuário enquanto o servidor fica livre para atender outros clientes. Da mesma forma, uma consulta a um banco de dados pode ser formulada através de recursos gráficos do cliente e sua execução se realizar no servidor ou com o auxílio do servidor. Enquanto a consulta é formulada ou enquanto seus resultados são apresentados e analisados no cliente, o servidor pode atender outros clientes.

b) Os recursos computacionais do servidor são compartilhados entre os usuários.

c) A utilização da rede fica otimizada, já que a comunicação pode ser feita via uma linguagem de alto nível tipo SQL e não via sistemas de arquivos.

d) O modelo possui uma camada de abstração acima do Sistema Operacional, tornando a comunicação mais transparente e portável.

Especificamente no desenvolvimento de SGBDs, a plataforma cliente/servidor vem sendo bastante adotada [Butt91, Lamb91, Mits92, Deux91, Fran92]. De um modo geral, os SGBDs são divididos com o cliente tratando do programa de aplicação do usuário e o servidor com o gerente de acesso à base de dados.

Roussopoulos e Delis [Rous91], no contexto dos sistemas relacionais, apresentam o ganho em desempenho obtido no processamento de consultas para abordagens que exploram o paralelismo proporcionado pela distribuição dos dados entre o servidor e o cliente. Já no contexto do paradigma de orientação a objetos, DeWitt e outros [DeWi90c] argumentam que devido às características de complexidade de manipulação de dados em um SGBDOO e à exigência de interfaces gráficas das aplicações não convencionais, uma solução de SGBDOO centralizada não é mais aceitável. Entretanto, não existe um consenso quanto à melhor forma de distribuição/alocação de módulos de arquiteturas de SGBDOOs entre cliente ou servidor.

Nesse sentido, o trabalho de DeWitt [DeWi90c] analisa três abordagens para arquiteturas cliente/servidor, a saber: servidor de arquivos, de páginas e de objetos. Essas três abordagens aumentam em ordem crescente a complexidade do servidor e cada uma possui suas vantagens e desvantagens, não havendo um consenso sobre a abordagem mais indicada. A caracterização das três abordagens é apresentada a seguir de acordo com [DeWi90c]:

i) Servidor de objetos

No servidor de objetos a unidade de transferência entre o servidor e o cliente é o objeto. O servidor conhece o conceito de objeto e é capaz de executar métodos sobre objetos. Os sistemas Orion, GemStone e O_2 (primeira versão) adotam o modelo servidor de objetos.

Vantagens

Os métodos podem ser executados sobre os objetos tanto nos clientes como no servidor, possibilitando uma divisão do trabalho entre os dois. Deste modo, um método que seleciona um sub-conjunto de uma coleção pode ser executado no servidor sem que o conjunto inteiro seja transportado para a estação de trabalho. A transferência de objetos do servidor para o cliente simplifica o controle de concorrência pois o servidor conhece que objetos são acessados por cada aplicação. Com esse controle a implementação do bloqueio no nível de objetos torna-se quase imediata. O 'buffer' de objetos no cliente também permite a ocupação de um espaço otimizado sem o desperdício do armazenamento de toda a página.

Desvantagens

No pior caso, pode haver uma chamada remota de procedimento ('rpc- remote procedure call') para cada referência do cliente a um objeto. Para que o servidor realize uma transferência com mais de um objeto, ele terá que conhecer o mecanismo de agrupamento dos objetos (função do cliente) replicando assim o código do SGBDOO entre clientes e servidor. Esta arquitetura aumenta a complexidade do servidor pois este deverá ser capaz de executar métodos arbitrários do usuário. Existe ainda o problema de inconsistência entre os 'buffers' do cliente, servidor e do disco no nível de objetos, o que torna a execução de uma consulta sobre uma coleção de objetos espalhada entre os três 'buffers' uma tarefa complexa. A questão de objetos longos merece uma atenção especial para evitar que o objeto inteiro seja carregado no cliente quando a aplicação necessitar de apenas alguns bytes do objeto grande.

ii) Servidor de páginas

O servidor trata de páginas e desconhece a semântica dos objetos. O servidor provê serviços de controle de concorrência e reconstrução. Como exemplos desta solução podem ser citados o Observer, Exodus e a segunda versão do O_2.

Vantagens

A maior parte da complexidade do SGBDOO fica localizada na estação de trabalho do cliente onde existem os maiores números de ciclos de UCP disponíveis. Deve-se considerar que o custo da transferência de uma página ou um objeto é praticamente o mesmo e nessa transferência se o mecanismo de agrupamento for otimizado é provável que o cliente usufrua de outros objetos da página carregada. A sobrecarga do servidor é minimizada pois páginas inteiras são transferidas entre o cliente e o servidor, com isso o servidor consegue atender a um número maior de clientes.

Desvantagens

Como o servidor não suporta métodos, uma busca a ser realizada sobre uma coleção de objetos faz com que todas as páginas envolvidas no armazenamento da coleção sejam transferidas para o cliente. Entretanto cabe observar que o número de operações de entrada/saída necessários será idêntico ao caso do servidor de objetos, sendo minimizada apenas a transferência entre o cliente e o servidor e a ocupação do 'buffer' do cliente. Outro problema está na implementação do controle de concorrência no nível de objetos já que o servidor desconhece o conceito de objetos.

iii) Servidor de arquivos

O servidor usa um serviço de arquivos remoto do tipo NFS (Network File System) [Sun 90] para ler e escrever páginas de disco diretamente e provê serviços de controle de concorrência e reconstrução. O GemStone para Sun adota essa solução.

Vantagens

A sobrecarga do servidor é minimizada e cresce a velocidade de transferência de dados, pois estes seguem diretamente para o cliente sem ter que passar pelo servidor (utilizando o NFS). Ao adotar páginas na transferência de dados este servidor incorpora as vantagens do servidor de páginas.

Desvantagens

Além das desvantagens do servidor de páginas, as operações de escrita no NFS são lentas. Além disso, o pedido de leitura da página não passa pelo servidor e não há a possibilidade de no mesmo pedido enviar a leitura e o bloqueio da página requisitada, implicando no aumento da complexidade do controle de concorrência. Problema semelhante ocorre na coordenação da alocação de espaço em disco.

IV.3 A ARQUITETURA DO GEOTABA

A arquitetura do GEOTABA se enquadra na classe de arquiteturas centradas em linguagens de programação de banco de dados, segundo a classificação de [Catt91]. Nessas arquiteturas os SGBDs possuem uma linguagem de consulta e uma linguagem de programação onde ambas são executadas no ambiente do programa de aplicação compartilhando o mesmo sistema de tipos e espaço de dados. Assim como o GEOTABA, diversos outros sistemas se encontram nesta categoria de arquitetura e utilizam o paradigma da orientação a objetos na linguagem de programação do banco de dados, como por exemplo os sistemas O_2 [Deux91], Orion (Itasca) [Kim 90a], GemStone [Maie86, Butt91].

As arquiteturas de SGBDs tradicionalmente se preocupam com o desempenho no acesso aos dados da aplicação através do uso de técnicas para persistência dos dados independente do programa de aplicação, da utilização de linguagens eficientes de consulta de alto nível, além de técnicas otimizadas para a gerência da memória secundária e de transações. Segundo Bancilhon [Banc92] essa ênfase da tecnologia de Bancos de Dados em desempenho tornou a sua integração com a programação orientada a objetos uma tarefa desafiadora. Nesse sentido, a arquitetura do GEOTABA busca a obtenção de um bom desempenho através das técnicas empregadas na gerência dos objetos armazenados e na utilização de uma plataforma do tipo cliente/servidor com exploração de paralelismo no servidor de objetos.

IV.3.1 A arquitetura funcional do GEOTABA

A arquitetura funcional do GEOTABA é composta por módulos/gerentes que desempenham tarefas necessárias a todo o ciclo de vida do objeto dentro do SGBDOO, apresentados na Figura IV.1. O usuário do GEOTABA realiza a modelagem de sua aplicação através do modelo de objetos do GEOTABA e interage com o SGBDOO utilizando a linguagem DEMO (que implementa o modelo) para definir e manipular seus objetos [Mont92]. A linguagem DEMO é mapeada para a linguagem MANO, uma linguagem de programação orientada a objetos com persistência que possui um processador próprio o ProtoGEO [Lisb92] responsável pelo processamento de mensagens/métodos.

Para o processamento das linguagens DEMO e MANO, os diversos gerentes da arquitetura são utilizadods, e nem todos os objetos seguem a mesma seqüência de passos entre os diversos gerentes. De um modo geral, pode-se pensar na instanciação de um objeto da aplicação do usuário passando pelo gerente de objetos, onde é feita a gerência do esquema, passando em seguida para o gerente de transações e para o gerente de armazenamento que estabelecerá sua alocação na memória principal e posteriormente na memória secundária, com os diversos controles de concorrência e reconstrução.

A arquitetura do GEOTABA pode ser visualizada através de seis módulos centrais (Figura IV.1) que correspondem aos:

- **Gerente de Execução (GE)**, responsável pela troca de mensagens, identificando-as e fazendo a associação entre a mensagem, o objeto e o método a ser ativado.

- **Gerente de Objetos (GO)**, implementa os diferentes níveis de abstração dos objetos e os mapeamentos entre esses níveis. O GO possui classes predefinidas responsáveis pela implementação do metaesquema do modelo de objetos. Servem como exemplo as classes CLASSE, ATRIBUTO, MÉTODO e COLEÇÃO. É responsável, também, pelo processamento de consultas.

- **Gerente de Objetos Armazenados (GOA)**, responsável pela gerência e controle dos objetos em memória principal, secundária e terciária e pela sua recuperação através de identificadores internos. O GOA gerencia métodos de acesso e a organização física dos objetos armazenados. Um componente do GOA cuida da organização e acesso físico aos meios de armazenamento (GMA-Gerente de Meio de Armazenamento) e outro componente gerencia a memória primária (GMO-Gerente da Memória de Objetos). Além disso, o GOA possui um componente que dá suporte ao Gerente de Objetos no processamento de consultas através da avaliação de predicados.

- **Gerente de Uso (GU)**, que abrigará as diversas interfaces de utilização do GEOTABA. Conterá o processador da linguagem textual DEMO (para DEfinição e Manipulação de Objetos) que implementa totalmente o modelo de objetos. Esta linguagem pode ser usada para consultas diretas ou de modo embutido em linguagens de programação como C++. Será também desenvolvida uma linguagem visual (VISOAL) baseada em uma representação gráfica da base de objetos, os Diagramas de Objetos (DOs) [Mont91]. VISOAL fornecerá meios para a manipulação direta da informação através de mecanismos de folheamento, hipertexto e consultas associativas.

- **Gerente de Interadores com Usuário (GI)**, que contém os objetos básicos para a apresentação de informação e interação com o usuário. Um de seus componentes cuida da utilização dos meios de interface com o usuário.

- **Gerente de Transações (GT)**, que é o responsável pelos controles operacionais do GEOTABA cuidando do compartilhamento da informação e concorrência de uso, da reconstrução, integridade e segurança da informação, além dos meios para gerenciamento de transações longas.

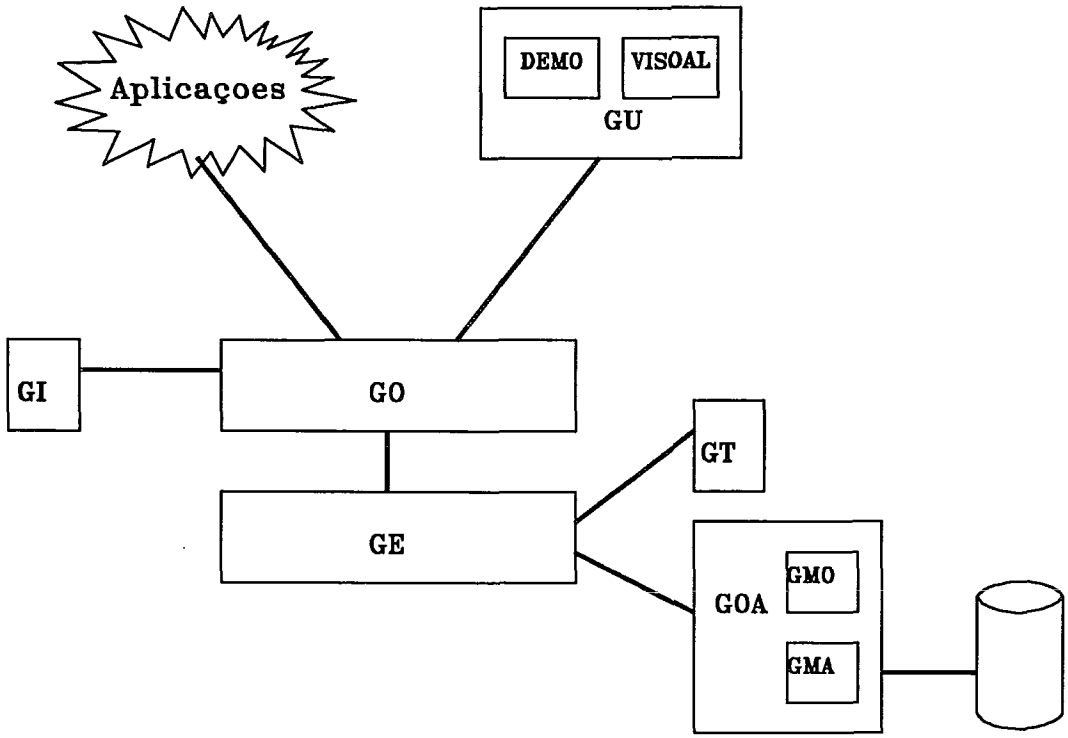

Figura IV.1 - Arquitetura do GEOTABA

IV.3.2 O GEOTABA na arquitetura cliente/servidor

A arquitetura do geotaba foi projetada para ambientes de estações de trabalho, com a plataforma cliente/servidor. Foi adotada a divisão do geotaba em cliente/servidor devido às diversas vantagens que essa arquitetura proporciona em termos de desempenho para o SGBDOO, além da facilidade de implementação da comunicação entre processos em ambientes de estação de trabalho Unix.

A comunicação entre os processos cliente e servidor do geotaba foi implementada através de "chamadas remota a procedimentos" ('rpc calls') utilizando a ferramenta 'RPC Gen' da Sun [Sun 90]. Detalhes dessa implementação podem ser encontrados em [Matt93].

Numa visão geral, a arquitetura do geotaba é composta de um servidor que mantém o controle centralizado da base de objetos armazenados e se comunica com os diversos clientes (usuários) que compartilham a mesma base de objetos, conforme Figura IV.2. A divisão dos módulos da arquitetura do GEOTABA em cliente/servidor adota o modelo do servidor de objetos com características semelhantes ao sistema Orion [Kim 90b].

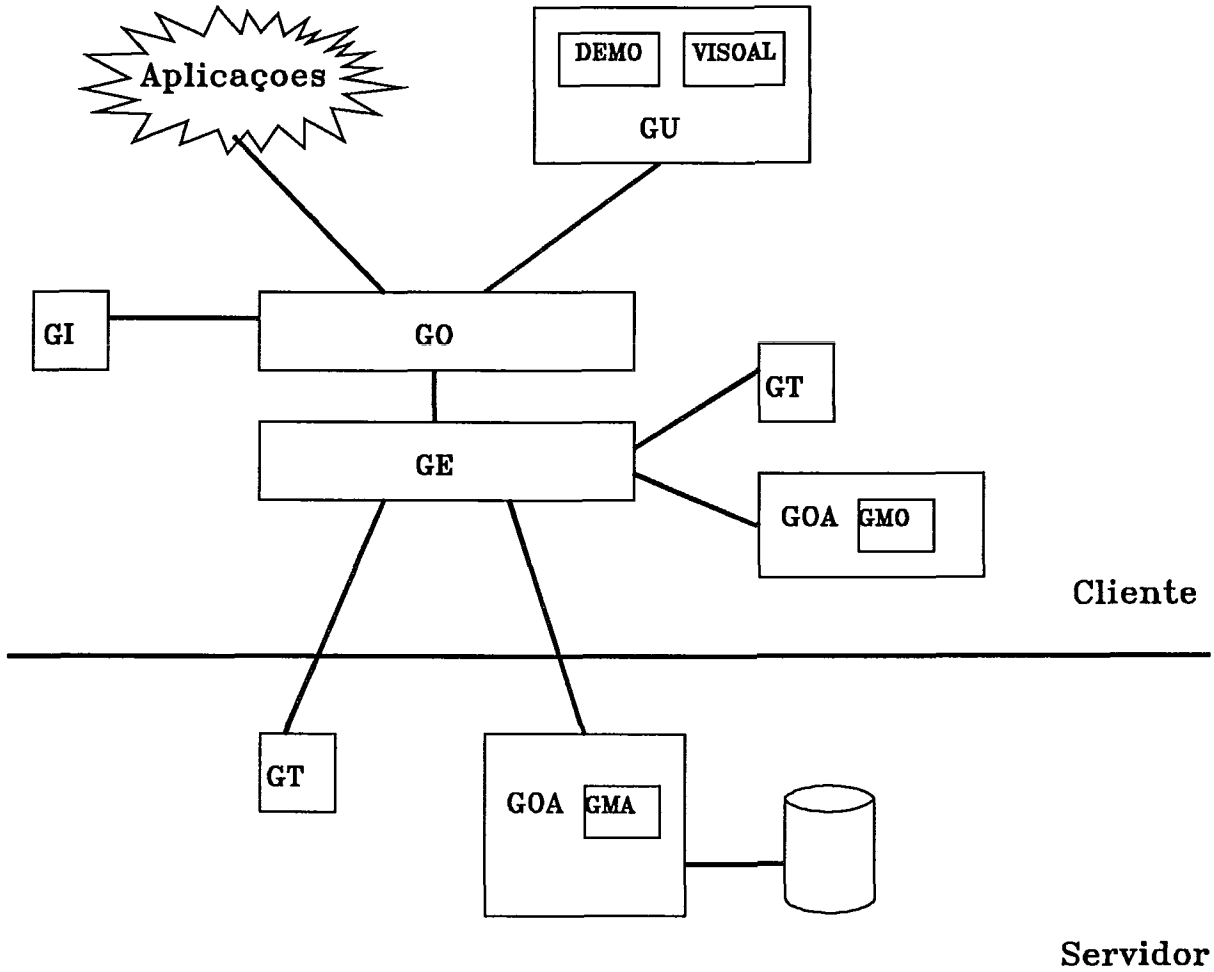

Figura IV.2 - Arquitetura cliente/servidor do GEOTABA

CLIENTE

No cliente, o usuário explora todos os recursos gráficos da estação para sua aplicação e trabalha com os gerentes de uso, de objetos e interadores com usuário (GU, GO e GI). Os demais gerentes ficam parcialmente no servidor e parcialmente no cliente, alguns com redundância.

A parte do GOA situada no Cliente corresponde ao GMO responsável pela gerência da memória de objetos que o usuário "dono" do processo Cliente, está trabalhando. O Cliente gerencia sua memória de modo semelhante aos processadores de linguagens de programação orientadas a objetos, ou seja não há preocupação com a gerência de páginas de disco. Cabe ao servidor garantir a persistência dos objetos enviados pelo cliente.

O Gerente de Transações (GT) reside, parcialmente, no Cliente com a tarefa de gerenciar algumas tabelas locais de bloqueio que são consultadas antes

do Cliente fazer uma requisição de bloqueio ao Servidor para o Gerente de Transações propriamente dito.

SERVIDOR

O Servidor contém o Gerente de Transações(GT) e o Gerente de Objetos Armazenados(GOA). Adota a solução do **servidor de objetos**, ou seja, a unidade de transferência de informação entre o cliente e o servidor é a de objetos e não páginas ou acesso direto a arquivos. Sendo assim, o GT pode utilizar o objeto para o seu nível de granularidade nas tabelas de bloqueio. O GOA no Servidor é responsável pela gerência de objetos armazenados no disco (agrupamento, gerência de coleções, etc.) e pelo controle do acesso às páginas do disco através da gerência do 'buffer'de páginas.

Além dos gerentes GT e GOA, o Servidor também dá suporte ao processamento de consultas do Gerente de Objetos no Cliente, através da avaliação de predicados no Servidor. O servidor de objetos tem capacidade de interpretar os objetos armazenados viabilizando assim o acesso associativo aos objetos, seja através de índices ou não.

Conforme apresentado na Seção IV.2.1, a escolha do tipo de servidor a ser adotada não é trivial, pois todas as opções oferecem vantagens e desvantagens. Foi adotada a solução do **servidor de objetos**, devido às vantagens de otimização na utilização do espaço de memória do cliente aliado ao aumento do nível de concorrência entre os clientes por conta da baixa granularidade de bloqueio. Quanto às desvantagens diversas soluções são propostas na literatura principalmente no que diz respeito à inconsistencia entre os 'buffers' [Kim 90a, Care91, Wang91, Wilk90]. Entretanto, o fator decisivo para a escolha do servidor de objetos foi a sua capacidade de conhecimento de parte da semântica do objeto, permitindo assim o acesso associativo aos objetos. Esta capacidade, possibilita a realização do processamento de consultas no âmbito do servidor, viabilizando assim o processamento de consultas através de um **servidor paralelo de objetos**. Desta forma, são concentradas no servidor as operações que possuem fontes de paralelismo a serem exploradas. Pretende-se que o servidor de objetos construído sobre a máquina paralela NCP I da COPPE proporcione alto desempenho para o GEOTABA. As questões de paralelismo no Servidor de Objetos serão abordadas no capítulo VI.

IV.3.3 Aspectos de implementação da arquitetura do GEOTABA

A arquitetura do GEOTABA pode ser vista de forma simplificada como uma **arquitetura em camadas**, correspondendo a três níveis de mapeamento entre os objetos do SGBDOO, conforme esquematizado na Figura IV.3. Na camada superior, encontra-se o Processador da Linguagem DEMO (**DE**finição e **M**anipulação de **O**bjetos), responsável pelo acesso e "interpretação" dos objetos no nível conceitual e pelo seu mapeamento para os objetos no nível de implementação. A camada do meio corresponde ao Processador da Linguagem MANO (**MAN**ipulação de **O**bjetos), que é equivalente a um processador de uma linguagem de programação orientada a objetos. A última camada manipula os objetos no nível interno através da sua representação na memória, gerenciando o mapeamento dos objetos armazenados entre as diversas memórias.

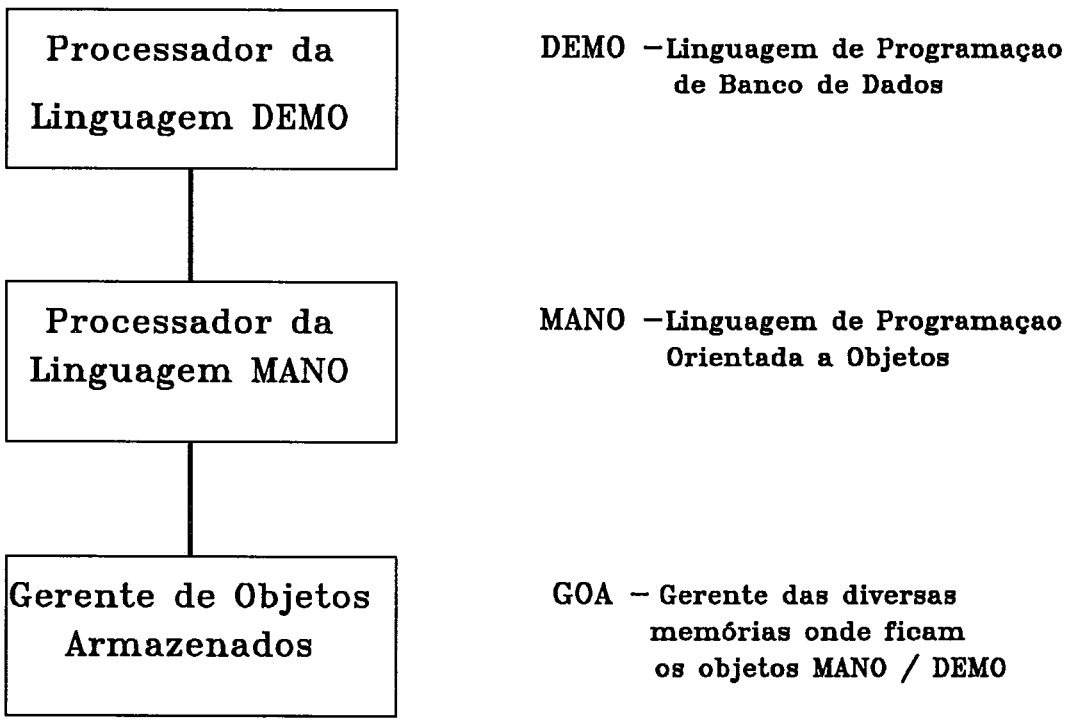

Figura IV.3 - Arquitetura em camadas do GEOTABA

Embora essas três camadas não sejam independentes, elas foram implementadas em separado prevendo sua integração e extensibilidade. Esses três módulos constituem a base de implementação da funcionalidade dos diversos gerentes da Figura IV.1.

O usuário submete ao GEOTABA, comandos na linguagem DEMO. Esses comandos são analisados pelo Processador DEMO gerando um código executável em MANO. Os comandos

em MANO são, então, analisados pelo Processador MANO que interage com o GOA para utilizar a gerência da memória.

O **Processador da linguagem de programação de banco de dados DEMO**, faz parte do trabalho de Monte [Mont93] e dá suporte à gerência do esquema do GO-Gerente de Objetos, ao GU-Gerente de Uso e ao GI-Gerente de Interadores com Usuário, gerando código executável através da linguagem MANO.

O **Processador da linguagem MANO** foi implementado por Lisboa [Lisb92] através do protótipo ProtoGEO e é responsável pelo processamento de mensagens/métodos escritos em MANO. O Processador da linguagem MANO dá suporte ao GE-Gerente de Execução.

O **Gerente de Objetos Armazenados GOA**, corresponde exatamente ao GOA da arquitetura funcional da Figura IV.1, sendo que o GMO (Gerente da Memória de Objetos) é implementado pelo ProtGEO, e as demais funcionalidades do GOA são implementadas pelo Servidor de Objetos do GOA. Desta forma, cabe ao Servidor de Objetos do GOA, a realização de um número predefinido de tarefas de gerência de objetos armazenados como, por exemplo, criação e remoção de objetos, gerência de coleções, listas, etc., além do acesso associativo a objetos de coleções.

Para gerenciar o armazenamento dos objetos em memória, o GOA adota o esquema de **'buffer' duplo** também utilizado nos sistemas O2, Orion e GemStone, onde o espaço de armazenamento é dividido entre um 'buffer' de páginas e um 'buffer' de objetos. Para que um objeto seja manipulado, a página que o contém deve ser trazida para o 'buffer' de páginas para em seguida ser feita a localização do objeto, sua obtenção e seu armazenamento no 'buffer' de objetos. Esse esquema de gerência de objetos tem sua origem no sistema LOOM [Kaeh83, Kaeh87] que propõe o uso do 'buffer' duplo para a gerência da memória virtual de linguagens de programação orientadas a objeto.

Segundo Kim [Kim 90a], as técnicas apresentadas por Kaehler tornaram-se a base dos esforços da integração das LPOO e os sistemas de banco de dados. Os objetos no 'buffer' de páginas estão no chamado formato de disco e no 'buffer' de objetos estão em seu formato de memória. Uma das vantagens desse esquema é que o programa de aplicação pode acessar diretamente os objetos do 'buffer' de objetos e a desvantagem está justamente na conversão do objetos entre os dois formatos. A Figura IV.4 mostra uma visão estrutural dos espaços de armazenamento de objetos no GOA.

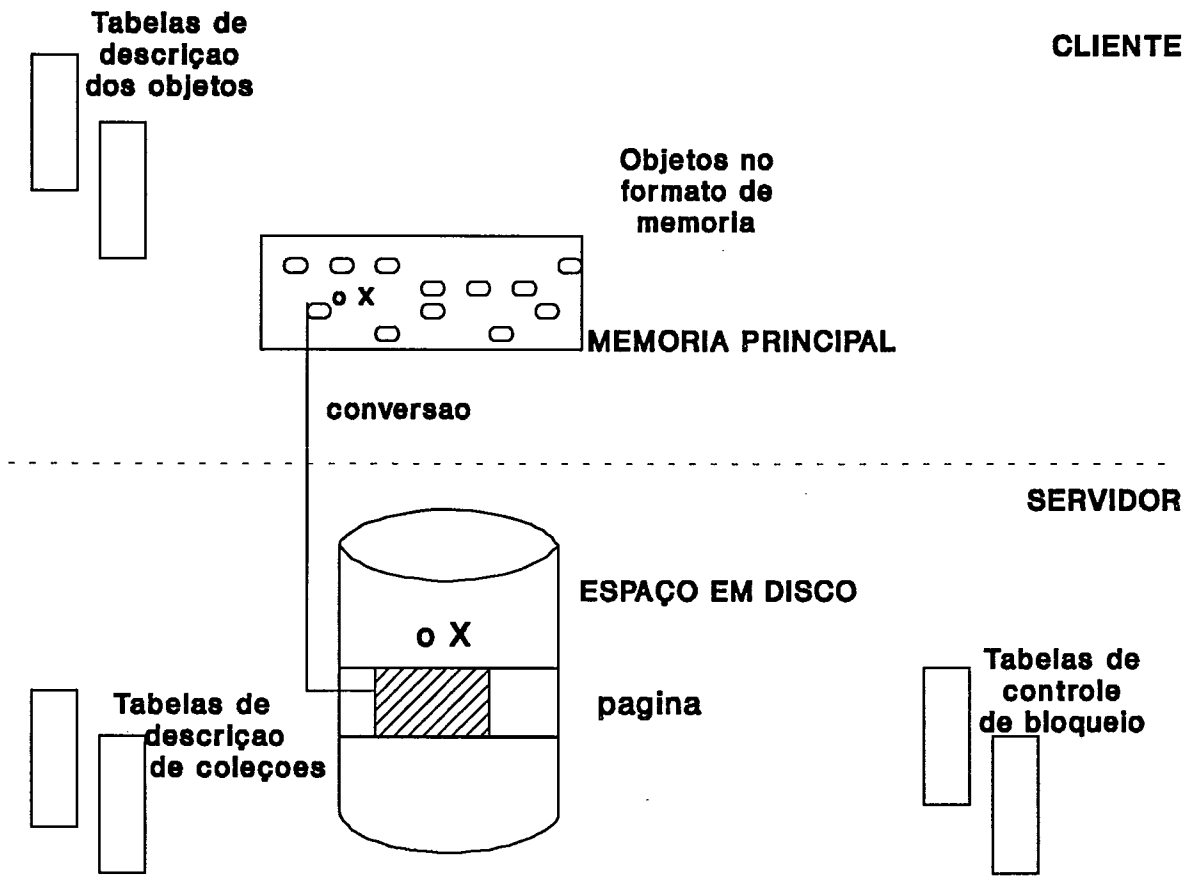

Figura IV.4 - Estruturas de armazenamento do GEOTABA

Embora diversos SGBDOOs utilizem o 'buffer' duplo para a gerência da memória virtual, a sua implementação varia de acordo com a distribuição do SGBDOO entre cliente e servidor. Conforme apresentado na Figura IV.4, o GOA dispõe de tabelas que implementam estruturas de dados que contribuem para a gerência eficiente dos objetos no 'buffer' de objetos e para o controle do uso concorrente desses objetos no espaço de armazenamento do cliente. O GOA, o Orion [Kim 90b] e o GemStone [Butt91] adotam o modelo do servidor de objetos e não necessitam manter um 'buffer' de páginas no cliente otimizando o uso do espaço de armazenamento do cliente. Já o sistema O_2 [Deux91] necessita manter um 'buffer' de páginas em cada cliente, mas por outro lado poderá tirar proveito da leitura antecipada dos outros objetos já existentes no 'buffer' de páginas do cliente.

A transferência de objetos entre cliente e servidor no GOA é realizada através da conversão entre o formato do objeto na memória principal (cliente) e o formato do objeto no disco (servidor) conforme mostra a Figura IV.5, que por sua vez corresponde à distribuição das estruturas da Figura IV.4. Solução semelhante é encontrada em [Chen91] e [Kim 90b].

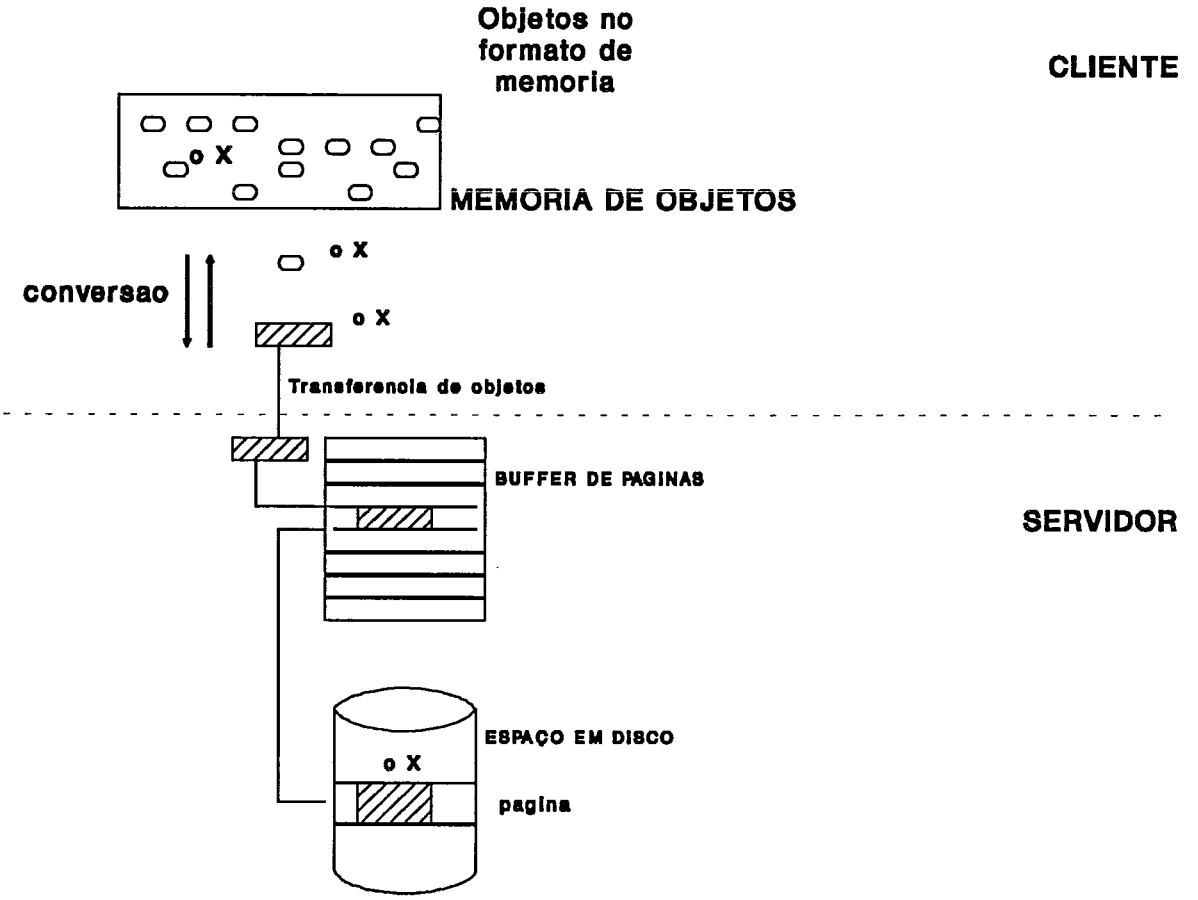

Figura IV.5 - Transferência de objetos entre cliente/servidor no GEOTABA

Quando a linguagem DEMO faz referência a um determinado objeto o processador DEMO se comunica com o processador Mano que precisa trazer o objeto para o 'buffer' de objetos. Essa operação segue uma seqüência de passos descrita de modo simplificado a seguir.

Leitura de objeto

Inicialmente o Gerente da Memória de Objetos (GMO) verifica se o objeto já está na memória, caso contrário ele envia uma mensagem ao Servidor de Objetos do GOA solicitando o objeto X. O Servidor, através do GMA lê a página que contém o objeto do disco, caso ela não se encontre no 'buffer' de páginas do servidor. Em seguida o servidor obtém o objeto X a partir do 'buffer' de páginas. O Objeto X é então envelopado e enviado via 'rpc' para o cliente que o recebe e faz a conversão do formato de disco para o formato em memória.

Gravação de objeto

De modo análogo, um objeto criado ou modificado no 'buffer' de objetos precisa ser gravado no disco. Inicialmente, o objeto necessita ser convertido pelo cliente do formato de

memória para o formato em disco. Em seguida, o objeto é envelopado e enviado ao servidor via 'rpc' solicitando seu armazenamento. No servidor a página que conterá o objeto é lida do disco para o 'buffer' de páginas (se necessário). Em seguida o objeto é armazenado na página do 'buffer' de páginas. Caso o objeto tenha aumentado de tamanho, o servidor aloca outra área para armazenar o objeto, preferencialmente na mesma página.

O servidor de objetos do gerente GOA é um dos focos centrais deste trabalho e suas estratégias de gerência são detalhadas no capítulo V.

IV.3.4 Os modos de operação da arquitetura do GEOTABA

As características de implementação da arquitetura do GEOTABA permitem três formas de distribuição descritas a seguir:

a) Totalmente Integrado

Nesta versão da arquitetura todos os módulos do GEOTABA são carregados simultaneamente na estação de trabalho, pronto para execução. Esse esquema só permite que um usuário se comunique com o sistema e toda a comunicação entre os módulos é feita via chamada de procedimentos.

b) Cliente/Servidor

Nesta versão, o servidor tem que ser carregado explicitamente na estação servidora, para que em seguida sejam carregados os processos clientes que se comunicam com o servidor. Esse esquema funciona sobre uma rede de estações de trabalho Sun com vários processos cliente se comunicando com o processo servidor via protocolo RPC (Remote Procedure Call) da Sun. Essa versão funciona de modo semelhante ao sistema O_2 onde o servidor atende seqüencialmente a uma fila de requisições dos clientes. Uma vez tendo sido carregado o servidor e o cliente pelo administrador do sistema, para o usuário esta comunicação fica transparente.

c) Cliente/Servidor Paralelo

Esta versão possui a mesma divisão da versão anterior com a diferença que o servidor é explicitamente carregado na máquina paralela NCP I que também está integrada à rede de estações Sun do ambiente computacional do GEOTABA. Os diversos clientes são disparados sobre estações de trabalho Sun e se comunicam com o servidor via protocolo TCP/IP [Sun 90].

IV.3.5 Características do Projeto GEOTABA

Para a implementação dos módulos da arquitetura do GEOTABA, foram desenvolvidos três trabalhos. O trabalho de Monte [Monte93], ficou responsável pelo Processador da linguagem DEMO. O trabalho de Lisboa [Lisb92] foi realizado através da implementação do ProtoGEO que dá suporte ao Gerente de Execução através de seu Processador MANO. O ProtoGEO também dá suporte ao Gerente da Memória de Objetos do GOA através de uma versão simplificada da gerência da memória de objetos. Desta forma, esses dois trabalhos, cuidam da implementação dos módulos do cliente da arquitetura (Figura IV.2). Finalmente, o terceiro trabalho situa-se no contexto desta tese, através da implementação do Servidor de Objetos do GOA que encontra-se descrita no próximo capítulo.

Para que os módulos do cliente e do servidor do GEOTABA estejam totalmente integrados, é necessário que o Gerente de Execução atualmente implementado no ProtoGEO, se comunique com o Servidor de Objetos do GOA. O ProtoGEO foi escrito em C++, estando em funcionamento em ambiente compatível com PC AT. Na implementação atual do ProtoGEO, a gerência da memória principal de objetos é realizada sem mecanismos de gerência de um espaço virtual. Entretanto, o ProtoGEO foi projetado para funcionar em sintonia com o GOA prevendo a gerência da memória virtual [Mont92].

Sendo assim, o ProtoGEO está sendo transportado para estações Sun onde fará uso da memória virtual do GOA e se comunicará via 'rpc' com o GMA-Gerente de Meios de Armazenamento do Servidor de Objetos do GOA para realizar as operações que envolvem persistência e acesso aos objetos armazenados. Uma das principais vantagens da utilização do ProtoGEO na arquitetura do GEOTABA está no processamento de sua linguagem de programação orientada a objetos que viabiliza um projeto independente de hardware e sem os limites de uma linguagem de programação orientada a objetos tipo C++ ou Smalltalk.

Capítulo V

O SERVIDOR DO GERENTE DE OBJETOS ARMAZENADOS

V.1 PRELIMINARES

O Servidor do GOA - Gerente de Objetos Armazenados, é o módulo responsável pela implementação das funções de gerência de objetos armazenados no servidor da arquitetura do GEOTABA (Figura IV.2). Este capítulo descreve os aspectos do desenvolvimento do Servidor do GOA, que realiza um número predefinido de tarefas de gerência de objetos armazenados como, por exemplo, alocação e remoção de objetos, gerência de coleções, listas, etc., além do acesso associativo a objetos de coleções. O servidor de objetos foi implementado em estações Sun e na máquina paralela NCP I. Antes da implementação da versão paralela do servidor GOA foi realizada a implementação seqüencial no sentido de estudar e avaliar técnicas de orientação a objetos para armazenamento de objetos e avaliação de consultas. A descrição do desenvolvimento do servidor GOA é realizada através de comentários em relação às técnicas apresentadas em [Catt91] e através de comparações com as implementações de outros SGBDOOs, em particular os sistemas O_2 e Orion. A implementação do processador de predicados de consulta no servidor de objetos mereceu uma Seção a parte pelo fato do processamento de consultas na orientação a objetos ser um tema atual de pesquisa onde existem diversas abordagens de implementação.

V.2 A GERÊNCIA DO ESPAÇO DE ARMAZENAMENTO

Devido ao esquema de 'buffer' duplo adotado (ver Seção IV.3.4), o GOA faz a gerência de três áreas de armazenamento: a memória de objetos, os 'buffers' de páginas e o espaço do disco. Com o mecanismo de servidor de objetos adotado, o servidor do GOA recebe os objetos já no formato de disco, isto é, como o servidor só gerencia sua memória virtual através do 'buffer' de páginas, o objeto já vem no formato em que ele é armazenado nas páginas e consequentemente no disco. Assim, o servidor GOA não precisa se preocupar com a gerência da memória de objetos.

Para implementar a gerência de páginas, segmentos e de disco, foram aproveitados e adaptados módulos do SGBD relacional COPPEREL [Matt87]. Essa adaptação e a característica de extensibilidade na construção do GEOTABA deu origem a uma biblioteca de rotinas, o sistema

SACO - Sistema de Armazenamento da COPPE [Matt93a] contendo uma série de primitivas para as seguintes gerências:

a) criação, abertura e fechamento da base de dados;

b) alocação e liberação de espaço em disco;

c) leitura e gravação de páginas via 'buffers';

d) leitura e gravação de registros sobre páginas;

e) acesso a registros via índices de espalhamento ('hashing').

Espaço em Disco

O espaço em disco é gerenciado na forma de volumes. Inicialmente é alocado um volume - uma grande área de disco sob a forma de um arquivo no nível do sistema operacional. Esse volume é logicamente dividido em páginas que constiuem a unidade de transferência entre o disco e a memória principal e vice-versa.

Segmentos

Os segmentos constituem uma área no disco que ocupa cinco páginas contíguas. Existem estruturas de dados que controlam a alocação e liberação de espaço dentro de um segmento além da ligação lógica de diversos segmentos para armazenar uma coleção por exemplo.

Buffer de Páginas

O 'buffer' de páginas é uma área em memória principal implementado de modo semelhante aos 'buffers' dos sistemas relacionais. No caso do GOA foram aproveitadas as rotinas de gerência do 'buffer' de páginas do COPPEREL através do sistema SACO que adota a política de 'NUR (Not Used Recently)' para a substituição de páginas no 'buffer'.

V.3 ESTRUTURAS DE ARMAZENAMENTO

Segundo Kim [Kim 90a], estruturas de armazenamento para SGBDOOs necessitam acessar eficientemente tanto os objetos simples, através de seus identificadores, quanto conjuntos de objetos pertencendo a determinadas classes, onde consultas complexas sobre atributos arbitrários possam ser realizadas.

i) Identificador do Objeto

Cattell [Catt91] apresenta os identificadores de objetos (IDO) como uma das principais características de um SGBDOO e uma das vantagens das implementações dos SGBDOOs em relação aos SGBDs relacionais está no uso de identificadores para representar referências a outros objetos ao invés do uso de valores (muitas vezes artificiais) adotado nos sistemas relacionais. Entretanto, a verificação de unicidade de objetos fica dificultada. Ainda em [Catt91] encontra-se uma classificação quanto aos diversos tipos de identificadores encontrados nos SGBDOOs. Segundo Cattell a técnica utilizada pelo SGBDOO para implementar o identificador dos objetos tem grande repercussão no desempenho do sistema.

Sendo assim, dentre as diversas maneiras de implementar um identificador de objeto, o GOA adota a solução de endereço estruturado. Esta solução também é chamada de endereço físico uma vez que parte do identificador contém o endereço da página "física" do armazenamento no disco e a outra parte contém o deslocamento dentro da página. O termo estruturado dessa solução vem do fato do endereço da página estar ligado a parte física enquanto que o posicionamento dentro da página pode ser feito indiretamente através de ponteiros ('slots') para os endereços dentro da página. Devido às características atuais do endereçamento do sistema SACO, não foi utilizada a técnica de 'slots' que será incorporada em versões futuras.

A principal vantagem do IDO físico está no seu desempenho que evita o acesso a uma tabela de objetos para a obtenção dos endereços físicos. Além disso, uma tabela para endereçar todos os objetos de uma base de dados pode se tornar difícil de gerenciar e provavelmente terá que ser armazenada no disco. Desta forma, a obtenção do objeto poderá ter um acesso ao disco para leitura da tabela e outro acesso para a leitura do objeto. Isto sem falar no problema do controle da concorrência uma vez que a tabela se torna um recurso muito compartilhado, principalmente quando se usa vários processadores no acesso à base de dados.

A implementação do IDO estruturado faz com que o objeto seja obtido na maioria das vezes com um único acesso a uma página no disco e no pior caso com dois acessos através da técnica de **endereço a seguir** ('foward address'). Com essa técnica, toda vez que um objeto precisa ser armazenado em outro endereço físico, o endereço original passa a ser usado como um ponteiro para o próximo endereço. A desvantagem desta técnica está no armazenamento de diversos ponteiros no disco. Esse endereçamento é também utilizado nos sistemas O2 [Vele89], Ontos e Objectivity/DB [Catt91].

Outro ponto favorável à utilização do IDO estruturado está na versão paralela do servidor do GOA, já que o ideal é que os processadores trabalhem de forma mais independente possível. A solução do identificador lógico, faz com que a tabela de objetos se torne um recurso altamente compartilhado e difícil de gerenciar.

ii) Formato dos Objetos no Disco

A representação adotada para o armazenamento dos objetos no disco é semelhante ao formato utilizado no sistema Orion [Kim 90b]. O formato dos objetos manipulados no servidor GOA pode ser visualizado na Figura V.1. O TAM OBJ corresponde ao comprimento do objeto, isto é, o tamanho que a representação total do objeto ocupa, incluindo os descritores e os valores dos atributos. CLASSE contém o IDO da classe a que o objeto pertence. O ATR# corresponde ao número de atributos armazenados. O ATR ID é um vetor dos identificadores dos atributos que foram instanciados neste objeto. O ATR DESL possui os respectivos deslocamentos para os valores dos atributos definidos no ATR ID. Finalmente, VALORES é a área que contém os valores definidos para cada atributo de ATR ID. A representação de atributos é discutida no sub-item *iii)* a seguir.

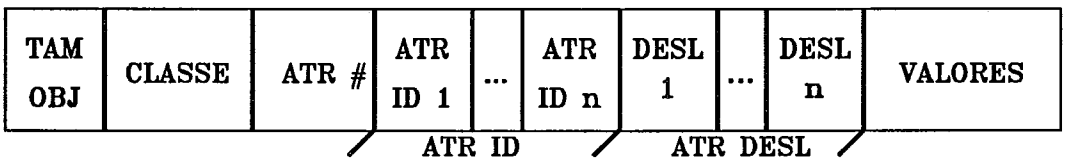

Figura V.1 - Formato de objetos no disco

Embora esse formato carregue alguma redundância quanto às informações do esquema, ele apresenta como vantagens a flexibilidade no armazenamento apenas dos atributos que o usuário escolheu para instanciar o objeto, além de permitir o tamanho variável para o valor de um determinado atributo estre os diversos objetos da classe. Cabe observar ainda que esse formato facilita a gerência da semântica dos objetos no âmbito do servidor, evitando que as tabelas do esquema sejam compartilhadas entre cliente e servidor. Este fato é ainda mais atraente para a versão paralela do servidor do GOA, onde diversos processadores podem avaliar predicados de consulta utilizando apenas as informações contidas no objeto armazenado.

iii) Representação de Atributos

Atributos Simples

São propriedades de objeto definidas como "literais". Para representar esses atributos é utilizado o espaço reservado para valores no formato de armazenamento do objeto onde o próprio valor do atributo é armazenado diretamente.

Atributos de Referência

São propriedades do objeto, onde seu valor faz parte da instânciação de outro objeto. Desta forma no espaço reservado para valores no formato de armazenamento do objeto encontra-se o IDO que identifica o objeto, e que permitirá o acesso ao valor desejado.

Atributos de Conjunto

São propriedades do objeto multivaloradas. Neste caso, no espaço reservado para valores no formato de armazenamento do objeto encontra-se o IDO do objeto tipo "lista" ou "conjunto" que contém o conjunto de valores atribuídos para o atributo de conjunto do objeto. Pode ocorrer do atributo de conjunto do objeto ser um conjunto de referências a outros objetos. Entretanto, a representação é a mesma de um conjunto de valores, isto é, o atributo conterá o identificador do objeto da classe conjunto que conterá os IDOs referenciados.

Atributos Longos

Embora a implementação atual do GOA não disponha de mecanismos especiais para a representação de atributos longos, sua incorporação será simples uma vez que o sistema de onde foram adaptadas as primitivas de acesso às páginas de disco já possui algoritmos de gerência de campos longos [Trot89] baseados em árvores-B segundo definição do sistema Exodus [Care86].

iv) Estruturas de Representação

O servidor GOA mantém uma série de estruturas/construtores que visam facilitar o armazenamento das instâncias das classes definidas no esquema do SGBDOO. Esses construtores facilitam a implementação dos atributos multivalorados e conjunto de objetos.

Listas

O construtor do tipo lista implementado visa suportar o armazenamento de objetos da classe *lista* (do modelo de objetos) e a implementação de estruturas de armazenamento mais complexas. A implementação de listas no GOA, pode ser vista como uma tabela de uma ou mais colunas, linearizada conforme o formato da Figura V.2, que segue o formato geral de objetos da Figura V.1.

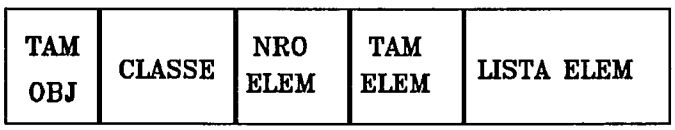

Figura V.2 - Formato de representação de listas

TAM OBJ contém o tamanho do espaço ocupado para a representação da lista. CLASSE indica a classe a que o objeto pertence, no caso a classe *lista*. Em NRO ELEM é armazenada a cardinalidade da lista, ou seja, o número de elementos inseridos na lista. TAM ELEM representa o espaço gasto por cada elemento, o que implica atualmente num formato fixo para a representação dos elementos da lista. Finalmente, em LISTA ELEM está a seqüência de valores para todos os elementos da lista. O servidor do GOA dispõe também de uma série de operações que manipulam listas, como por exemplo a obtenção do i-ésimo elemento, ou

ainda no caso de uma tabela a obtenção/modificação da j-ésima coluna do i-ésimo elemento. Entretanto, o controle de tipos ou tamanho de colunas da lista é realizado externamente ao construtor. A documentação dos construtores pode ser encontrada em [Matt93a].

Coleções

O construtor do tipo coleção visa dar suporte ao mapeamento dos objetos que pertencem a uma classe que é do tipo *coleção*, conforme especificado no modelo de objetos [Mont93]. Uma coleção no GEOTABA tem a semântica semelhante ao construtor 'named set' do sistema O_2[O_2Te]. O construtor coleção tem como função principal o armazenamento dos IDOs dos objetos que pertencem a uma coleção e não necessariamente pertencem à mesma classe. Isto é, assim como no O_2, uma coleção pode não corresponder à extensão da classe a ela associada.

Para implementar o construtor coleção, o GOA dispõe de uma TABELA DE COLEÇÕES que descreve a coleção através de seu nome, número de objetos inseridos, a lista dos IDOs dos objetos e uma lista dos segmentos que armazenam os objetos propriamente ditos, de acordo com a técnica de agrupamento sendo utilizada. Nem todos os objetos que pertencem a uma coleção são armazenaddos contiguamente ou próximos. Os detalhes da especificação e implementação do construtor coleção estão descritos em [Matt93a].

v) Agrupamento

Cattell [Catt91] e Kim [Kim 90a] também consideram o agrupamento de objetos no disco como um dos fatores que tem impacto substancial no desempenho de um SGBDOO. Alguns trabalhos [Tsan91 e 92] avaliam impactos do agrupamento através de análises de diversos algoritmos. Tsangaris e Naughton apostam no agrupamento estocástico que leva em conta as seqüências de acesso aos objetos visando a distribuição ótima dos objetos entre as páginas. O agrupamento dos objetos num SGBDOO pode ser realizado de diversas formas. O servidor do GOA adota mecanismos simples de agrupamento nesta primeira versão. Utilizando a classificação de agrupamento de Chang [Chan89], os agrupamentos suportados no GOA são os de :

. *Composição de objetos* - Objetos que estão relacionados via agregação são armazenados juntos. Desta forma são armazenados próximos objetos que se relacionam explicitamente por composição e os atributos de um mesmo objeto que ocorrem na definição da hierarquia da classe. Atributos do tipo lista, também consistem de um objeto em si e são armazenados junto ao objeto que os referencia.

. *Objetos do mesmo tipo* - Nessa agregação objetos de uma mesma classe ou tipo são armazenados contiguamente. Entretanto, esse agrupamento só é vantajoso se vários acessos são realizados aos objetos do mesmo tipo. Desta forma, o GOA utiliza uma variação desse agrupamento, só armazenando contiguamente os objetos que pertencem a

uma mesma coleção. Sabe-se que quando a aplicação modela uma classe através de coleções significa que existe a intenção do acesso associativo àqueles objetos.

. *Referências* - Nesse agrupamento, objetos que referenciam outros objetos são armazenados juntos. O GOA também utiliza uma variação deste agrupamento, onde só ficam juntos os objetos referenciados que são do tipo lista. Entretanto, considera-se que esse agrupamento já esteja contemplado na *composição de objetos*.

Pretende-se implementar numa próxima versão do GOA uma estratégia de armazenamento atualmente suportada pelo sistema WISS [Chou85], onde ao se armazenar um determinado registro r_i (objeto), o sistema oferece a opção de especificar o identificador de outro registro r_j da base de dados. O WISS tenta armazenar o novo registro r_j (objeto) na mesma página de r_i ou numa página próxima. Esse tipo de agrupamento é classificado por Chang [Chan89] como *adaptado* ('custom') e é utilizado nos sistemas O_2 e Objectivity/DB [Catt91].

O agrupamento atualmente implementado no GOA ocorre em dois níveis [Catt91]:

. Páginas - Essa unidade lógica de agrupamento é utilizada para o agrupamento de composição de objetos.

. Segmentos - Listas de segmentos são utilizadas para agrupar o armazenamento de objetos de uma mesma coleção.

O mecanismo do O_2 é sofisticado [Vele89] e parece trazer bons resultados dada a sua flexibilidade. Como sua implementação é de certa forma independente do sistema de armazenamento, pretende-se estudar a sua implementação em futuras versões do GOA.

V.4 AS OPERAÇÕES DO SERVIDOR

A seguir são apresentadas as operações atualmente implementadas no servidor do GOA. O cliente envia o código correspondente às operações e seus respectivos parâmetros de entrada via 'rpc'. O servidor identifica o código da operação, executa a rotina adequada e envia o resultado também via 'rpc' de volta para o cliente. O detalhamento das operações e seu pseudo-código encontra-se em [Matt93a].

1) Armazenar objeto novo.
2) Recuperar objeto.
3) Armazenar objeto modificado.
4) Armazenar objeto novo em uma coleção.
5) Inserir objeto, já armazenado na base, em uma coleção.
6) Remover objeto da coleção.

7) Procurar objeto em uma coleção.
8) Percorrer uma coleção.
9) Operações de consulta.
10) Atribuição de coleção.

V.5 O AVALIADOR DE CONSULTAS

Poucos resultados têm sido obtidos na definição e padronização de um modelo de consultas ou de linguagens de consulta [Catt91, Kim 90a], embora a maioria dos SGBDOOs disponham de uma linguagem e processador de consultas [Banc89, Kim 89, Care88]. Algumas propostas de definição formal para uma álgebra de objetos podem ser encontradas nos trabalhos de Straube e Oszü [Stra90] e Shaw e Zdonik [Shaw90].

Bancilhon [Banc89] apresenta características que uma linguagem de consulta deve ter e discute uma lista de aspectos que precisam ser definidos ao se projetar uma linguagem de consulta para um modelo de dados orientado a objetos, como por exemplo, o que deve ser o resultado de uma consulta, a questão do encapsulamento, entre outros. Moura [Mour92b] estende essa lista e discute outros aspectos como recursividade e o conceito de igualdade entre objetos no contexto de um predicado de consulta. Cattell [Catt91] também discute as novas questões introduzidas pela orientação a objetos no processamento de consultas e faz uma análise do ponto de vista da integração da linguagem de consulta e da linguagem de programação do SGBDOO.

A necessidade de uma ferramenta de consulta num SGBD é inegável, seja qual for seu modelo de dados. A utilização de uma linguagem de consulta em detrimento a uma linguagem de programação é encorajada por diversos autores na literatura [Kim 90a, Ston90, Catt91] por se tratar de uma linguagem de nível mais alto e passível de otimização por parte do SGBD. No caso do GOA paralelo, o uso de uma linguagem de consulta é ainda mais incentivado, pois nela se encontram as maiores fontes de paralelismo, conforme apresentado no capítulo VI.

Para que um SGBDOO disponha de uma ferramenta para acesso aleatório à base de dados e consultas 'ad-hoc' é necessário que os projetistas do SGBDOO desenvolvam:

- um modelo de consultas,
- uma linguagem de consulta,
- um analisador da linguagem de consulta (com otimização),e
- um processador de consultas.

V.5.1 Alternativas de Projeto do Avaliador de Consultas

Poucos sistemas apresentam, formalmente ou não, o modelo de consultas adotado no SGBDOO. Em geral, são apresentadas algumas características da linguagem, a sintaxe e exemplos. Poucos autores comentam sobre o processamento da consulta em si. Kim [Kim 90a] apresenta uma proposta de modelo genérico para consultas no Orion baseado em grafos. Uma estrutura de grafo é utilizada para representar as classes, subclasses e as classes dos domínios dos atributos envolvidos na consulta.

O modelo de consultas a ser adotado na construção de um SGBDOO está intimamente ligado ao modelo de objetos que o SGBDOO deverá suportar. Da mesma forma, a linguagem de consulta possui ligações com a linguagem de programação do banco de dados.

No caso do GEOTABA, o modelo de consultas é ligado ao modelo de objetos através da classe *coleção* e a linguagem de consulta é um sub-conjunto da linguagem DEMO. Conforme apresentado na arquitetura do GEOTABA (Seção IV.3.3), cabe ao GOA, no servidor de objetos, o suporte ao processamento de consultas através da avaliação de predicados. Desta forma, o servidor recebe como uma de suas operações, um código intermediário para realizar o processamento do predicado enviado pelo Gerente de Objetos. Este predicado foi previamente analisado pelo processador da linguagem DEMO e posteriormente analisado pelo otimizador de consultas do Gerente de Objetos.

Para que o processador de consultas pudesse ser desenvolvido em paralelo à especificação e ao projeto da linguagem DEMO, foi definida uma pseudo-sintaxe de cláusulas de qualificação de consultas. Desta forma, pode ser especificado um código intermediário das operações de avaliação de predicados implementadas na atual versão do servidor de objetos do GOA.

Sendo assim, a Sub-seção a seguir não aborda a questão de consultas em SGBDOOs, mas apresenta as soluções adotadas para o processamento de consultas no GEOTABA através da avaliação de predicados implementada no GOA.

V.5.2 Escopo e Resultados da Consulta

Assim como no Orion e no O_2, a linguagem de consulta do GOA permite que a população (objetos) de uma classe seja percorrida e acessada. Entretanto, o O_2 [O_2Te] e o GOA restringem o escopo da classe alvo da consulta para que a classe esteja associada a uma coleção, ou seja, o alvo de uma consulta no comando 'select' não pode ser uma classe qualquer. No caso do O_2, o alvo da consulta é sobre uma coleção nomeada ('named set'), persistente associada a uma determinada classe do esquema definida no momento da criação da coleção nomeada. Já no caso do GOA o alvo da consulta tem que ser sobre uma classe do tipo *coleção*, definida já na modelagem da aplicação.

Essa restrição não impede que classes que não estejam associadas a coleções sejam referenciadas ao longo do predicado da consulta. A restrição ocorre em respeito ao modelo de objetos adotado, onde uma classe não caracteriza um conjunto ou uma coleção de objetos. Desta forma, o acesso associativo a uma classe só faz sentido se essa classe possui um tratamento de conjunto ou de coleção para seus objetos, segundo a modelagem do usuário. Entretanto, instâncias de subclasses da hierarquia da classe correspondente à coleção alvo, também estarão envolvidas no acesso associativo, desde que essas instâncias pertençam à coleção alvo.

Já no sistema Orion [Kim 89], qualquer classe do esquema pode ser o alvo de uma consulta, pois a consulta à base de dados é realizada através do acesso à população da classe ("extensão"). No Orion, todos os objetos de uma classe são automaticamente gerenciados pelo sistema como uma coleção, mesmo que a classe tenha uma população pequena.

Quanto ao resultado da consulta, o Orion e O_2 adotam enfoques bastante distintos. No sistema O_2, como o modelo de objetos convive com objetos e valores, o resultado pode envolver uma combinação quase ilimitada de classes(atributos) para gerar o resultado da consulta como valores. Já o sistema Orion [Kim 90a], respeita o encapsulamento de objetos e o resultado da consulta fica limitado a IDOs que pertencerão a classes geradas automaticamente pelo sistema. Tal fato limita a combinação de atributos nos objetos resultantes, mas não impede que sejam realizadas junções entre classes uma vez que pode ser gerada uma hierarquia contendo as classes envolvidas.

V.5.3 A Pseudo-Sintaxe da Linguagem de Consulta

Foi adotado um modelo de consultas semelhante ao modelo do Orion [Kim 89] devido à sua compatibilidade com a arquitetura projetada para o GEOTABA e com o modelo interno de objetos do Gerente de Armazenamento. O fato do GOA utilizar um servidor de objetos com capacidade para processamento de consultas, fez com que as soluções do Orion se tornassem atraentes para as avaliações de predicado no GOA. Além disso, o fato do servidor não compartilhar das informações do esquema gerenciadas no cliente, fez com que a avaliação de consulta fosse quebrada em cliente e servidor, restringindo a capacidade de processamento no servidor.

As seguintes simplificações foram realizadas no modelo do Orion para a implementação corrente. As consultas são do tipo acíclicas, ou seja, não envolvem ciclos nem recursividade. Além disso, não foram implementadas cláusulas de predicados que restringem o seu escopo para subclasses especificadas dentro da hierarquia da classe da coleção alvo. O servidor retorna como resultado da avaliação do predicado para o avaliador da consulta no cliente a lista dos IDOs da classe alvo que qualificam o predicado.

Não houve preocupação com a elaboração de uma sintaxe amigável ou completa, uma vez que a linguagem de consulta do GEOTABA é um sub-conjunto da linguagem DEMO [Mont93] que está fora do escopo deste trabalho. No contexto do servidor de objetos do GOA, a consulta já é recebida em seu código intermediário.

A seguir é apresentada a pseudo-sintaxe das cláusulas de qualificação das consultas. Só será apresentada a sintaxe das cláusulas implementadas na atual versão do GOA. A sintaxe segue um formato tipo BNF com simplificações em relação à definição de alguns termos **não terminais** que não foram expandidos por serem auto-explicativos.

```
CláusulaQualificação ::=
    ( CaminhoEscalar ComparadorEscalar ElementoEscalar |
     CaminhoEscalar ComparadorEscalarConjunto ElementoConjunto |
     CaminhoConjunto ComparadorConjunto ElementoConjunto |
     CaminhoConjunto ComparadorConjuntoEscalar ElementoEscalar |
     CláusulaQualificação "e" CláusulaQualificação |
     CláusulaQualificação "ou" CláusulaQualificação ).

CaminhoEscalar  ::= (ElementoCaminhoEscalar |
                     ElementoCaminhoEscalar CaminhoEscalar).

ElementoCaminhoEscalar ::= ( NomeAtributoEscalar |
                             Quantificador NomeAtributoConjunto).

Quantificador ::= ("existe" | "para cada").
ElementoEscalar ::= ( ConstanteEscalar | IDO ).
ComparadorEscalar ::= ( "=" | ">" | "<" | ">=" | "<=" | "!=" ).
CaminhoConjunto ::= (ElementoCaminhoConjunto |
                     ElementoCaminhoConjunto CaminhoConjunto ).

ElementoCaminhoConjunto ::= (NomeAtributoConjunto |
                      NomeAtributoConjunto CaminhoEscalar|
                     CaminhoEscalar NomeAtributoConjunto).

ElementoConjunto ::= ( ConjuntoConstantes | NomeColeção ).
ComparadorEscalarConjunto ::= "is-in"
ComparadorConjuntoEscalar ::= "has-element"
ComparadorConjunto ::= "has-subset"
```

V.5.4 A Implementação do Processador de Consultas

A avaliação de predicados é implementada no GOA da mesma forma que as demais operações suportadas pelo servidor de objetos. O servidor identifica o código de **seleção** e a execução é desviada para o módulo de processamento de consultas. Para implementar os diversos tipos de consultas apresentados na sintaxe, foi especificado um algoritmo genérico utilizando o código intermediário apresentado a seguir.

O código intermediário é representado (com algumas simplificações) através de uma lista de elementos com a seguinte composição:

- **COD. OPERAÇÃO :** seleção.
- **TIPO COLEÇÃO :** código que identifica (coleção I IDO da lista de objetos de uma coleção).
- **COLEÇÃO :** nome da coleção.
- **#OU :** número de cláusulas "ou" ('or').
- **#E :** número de cláusulas "e" ('and').
- **OPERADOR :** código que identifica (">"I "="I "<"I ">="I "<="I "!="I "is-in"I "has-element"I "has-subset".
- **QUANTIFICADOR :** código que identifica ("existe" I "para cada").
- **#NÍVEIS :** número de atributos no caminho do operando.
- **LISTAATR :** lista com os nomes dos atributos.
- **TIPOATR :** tipo do atributo que será comparado.
- **CLASSEVALOR :** código que identifica o **tipo do valor** a ser comparado com o atributo (constante escalarI IDOI coleção de constantesI nome de coleção).
- **VALOR :** conteúdo ou valor a ser usado na comparação.

O algoritmo de avaliação dos predicados adota a estratégia descendente de resolução de consultas. Outras estratégias também foram experimentadas na implementação do GOA e são comentadas na próxima Seção. O detalhamento dos algoritmos pode ser encontrado em [Matt93a]. O algoritmo em linhas gerais, é apresentado como segue:

. Obtém lista de IDOs da coleção alvo.
. Se #e > 0, avalia cláusulas "e"
. Percorrer lista IDOs da coleção
    . Faça:
        . CONTADORDEES ++
        . Caso OPERADOR seja:
            "comparador escalares":
                . PegaValorAtributo;
                . ComparaValorAtributoComValorPredicado;
            "is-in":
                . Caso CLASSEVALOR seja:
                    "constante escalar"
                        . PegaValorAtributo;
                        . PercorreColeçãoConstantes;
                        .ComparaValorAtributoComValorConstante;
                    "nome de coleção"
                        . PegaValorAtributo;
                        . Percorre lista de IDOs da coleção;
                            .ComparaIDOAtributoComIDOColeção;
            "has-element":
                ...
            "has-subset":
                ...
    Até comparação Falsa ou CONTADORDEES = #e;
    . Se comparação Verdadeiro
        Então : RESULTADO <-- IDO corrente da coleção alvo.

V.5.5 Avaliação de Estratégias de Processamento de Consultas

O processamento de uma consulta no modelo orientado a objetos pode variar de acordo com o número de classes envolvidas na avaliação dos predicados. Diversas estratégias podem ser usadas na escolha do caminho a ser percorrido durante a avaliação. Assim como a junção de N relações pode ser feita através de N! permutações, N classes de uma hierarquia de composição podem sofrer "junções" sobre quaisquer das N! permutações, isto supondo uma consulta sobre uma única classe alvo.

Embora o servidor do GOA não possua ainda um otimizador de consultas, as técnicas de processamento de consultas têm grande influência no desempenho do SGBDOO. Visando dar subsídios à implementação do otimizador de consultas, foram analisadas diversas estratégias de avaliação de consultas no contexto da orientação a objetos, e especificamente no âmbito do GEOTABA.

Apesar da possibilidade de N! permutações para a avaliação de um predicado de consulta, serão analisadas três classes de estratégias a seguir, inicialmente, através de sua descrição e posteriomente com a discussão das questões de implementação e das vantagens e desvantagens.

a) Descendente

A estratégia descendente funciona de modo semelhante aos algoritmos de laços aninhados do modelo relacional. Esta estratégia percorre todos os objetos do caminho especificado na cláusula, descendo pelas classes dos atributos envolvidos até chegar ao último nível, voltando em seguida para a avaliação mais interna e assim sucessivamente.

b) Ascendente

Neste algoritmo, seriam percorridos inicialmente, os objetos da classe do último nível de atributos da cláusula de qualificação. Isto porque é neste último nível que são aplicadas as operações de comparação com a constante. Em seguida os IDOs dos objetos da classe que satisfazem a condição sofrem uma operação de "junção" com os atributos da classe imediatamente superior no caminho do predicado e assim sucessivamente. Para executar essa "junção" entre os IDOs podem ser usados os algoritmos de "laços aninhados" ou de 'sort-merge' desenvolvidos para o modelo relacional. O algoritmo termina com a junção dos objetos da classe alvo com a lista dos IDOs selecionados ascendentemente.

c) Ascendente/Descendente

Outra alternativa seria percorrer inicialmente o nível mais interno (o último nível) avaliando a condição de seleção para em seguida realizar o algoritmo descendente até o penúltimo nível e finalmente, realizar a junção com os IDOs selecionados. Dependendo do

número de objetos das classes envolvidas no predicado este algoritmo também pode seguir ascendentemente por alguns níveis e só então realizar o percurso descendente.

Uma das vantagens da utilização do percurso ascendente/descendente reside na possibilidade de diminuição do número de acessos ao disco. Quando a condição de seleção é restritiva, e quando o número de instâncias da classe do último nível do predicado é menor que o número de instâncias da classe alvo da consulta, obtem-se um ganho no desempenho face à otimização dos acessos ao disco. Já na estratégia totalmente ascendente, o ganho é mais difícil de ser projetado e também mais improvável de ocorrer, entretanto essa estratégia pode ser interessante para o processamento distribuído ou paralelo conforme analisado no capítulo VI.

Embora as estratégias que envolvam percurso ascendente possam oferecer ganhos em desempenho, a sua implementação não é imediata. A seguir são levantadas considerações a serem verificadas ao se escolher um algoritmo que envolve um percurso ascendente no âmbito do servidor do GOA.

O modelo de consultas adotado no GEOTABA restringe o escopo de consultas sobre coleções alvo de forma semelhante ao O_2. Isto é, o usuário não pode realizar consultas sobre os objetos de uma classe qualquer (já que a classe não caracteriza um repositório de objetos) mas sim sobre coleções de objetos. O modelo interno implementado também segue essa orientação. Existe um tratamento especial para os objetos que pertencem a determinada coleção. Em outras palavras, o servidor mantém uma lista de objetos para cada coleção definida pelo usuário. Essa lista não necessariamente coincide com **todos** os objetos da classe associada à coleção. Assim como no O_2, o usuário pode instanciar um objeto de uma classe e não desejar que ele pertença à coleção associada à classe.

Como conseqüência indireta, é necessário que se tenha acesso a todos os objetos de uma determinada classe, para se percorrer os objetos de uma classe de um nível qualquer da cláusula de qualificação (com excessão do primeiro nível), mesmo que essa classe não possua uma coleção associada. Isto não é viável no modelo interno atual adotado no GOA. Mesmo que a classe a ser percorrida possua uma coleção associada, esta coleção não poderá ser utilizada pois pode não coincidir com a população da classe. Desta forma, apesar de ter sido experimentado o algoritmo ascendente/descendente, somente a estratégia descendente está disponível para o cliente na versão atual do GOA. A próxima Seção analisa as implicações de introduzir efetivamente (e não experimentalmente conforme foi realizado) a estratégia de avaliação ascendente de predicados de consulta.

V.5.5.1 Avaliação da Implementação da Estratégia Ascendente

Para que o avaliador de consultas possa se beneficiar da estratégia ascendente, diversas mudanças devem ser realizadas no servidor do GOA. Para que seja possível percorrer os objetos de uma classe qualquer, o servidor precisa gerenciar uma estrutura tipo lista que contenha todos os IDOs dos objetos instanciados na classe. A manutenção de uma lista dessas para uma classe envolve os seguintes custos:

- gastos no espaço de armazenamento para uma atividade específica, que é a consulta;
- redundância, já que listas quase iguais serão mantidas nas coleções;
- as operações de inserção e remoção de objetos ficarão mais complexas e caras;
- haverá necessidade do servidor de objetos conhecer a hierarquia de classes, o que implica numa redundância difícil de ser controlada. Isto é, informações do esquema conceitual deverão estar presentes tanto no servidor quanto no cliente (gerente de armazenamento e gerente de objetos respectivamente). A não ser que o gerente de objetos já informe ao gerente de armazenamento, a que classes o objeto pertence, para que as inserções nas listas sejam realizadas. Idem para as remoções.

Conforme pode ser observado, a implementação das listas/coleções não é trivial. O custo da manutenção de listas para todas as classes nem sempre resultará em benefício pois determinadas classes poderão nunca estar envolvidas em consultas. Uma alternativa seria manter uma lista de objetos para determinadas classes e restringir a pesquisa ascendente a estas classes. Essa alternativa diminui os custos através da limitação do número de classes cuja lista de objetos será mantida. Entretanto, o sucesso dessa alternativa depende da sintonia do sistema no sentido de escolher as classes "certas" para possuirem o acesso via listas.

No sentido de contribuir para os possíveis ganhos proporcionados pela avaliação ascendente/descendente (A/D) de consultas, alguns testes foram realizados. Embora o modelo interno atual do GOA não suporte a avaliação ascendente, foram implementadas algumas listas experimentalmente para que diversas classes pudessem ser utilizadas na estratégia A/D. Quanto aos novos algoritmos, foram aproveitados os códigos dos predicados que envolvem os *comparadores* "is-in" e "has-subset".

Para a avaliação quantitativa da estratégia A/D, foi necessária a geração de uma base de dados para a realização dos testes. Embora Cattell [Catt92] tenha proposto um teste padrão ('benchmark') para operações sobre objetos, optou-se pelo aproveitamento dos dados já utilizados previamente. Foi medido o tempo de execução para consultas sobre uma base de dados hipotética com instâncias para a modelagem da aplicação da Figura V.3. As instâncias foram obtidas a partir de uma adaptação dos dados utilizados na avaliação do PARBASE (descritos no capítulo III). Os tempos obtidos foram comparados em relação à utilização da estratégia descendente e da

estratégia A/D. Os resultados quantitativos confirmaram as expectativas da análise qualitativa realizada anteriormente.

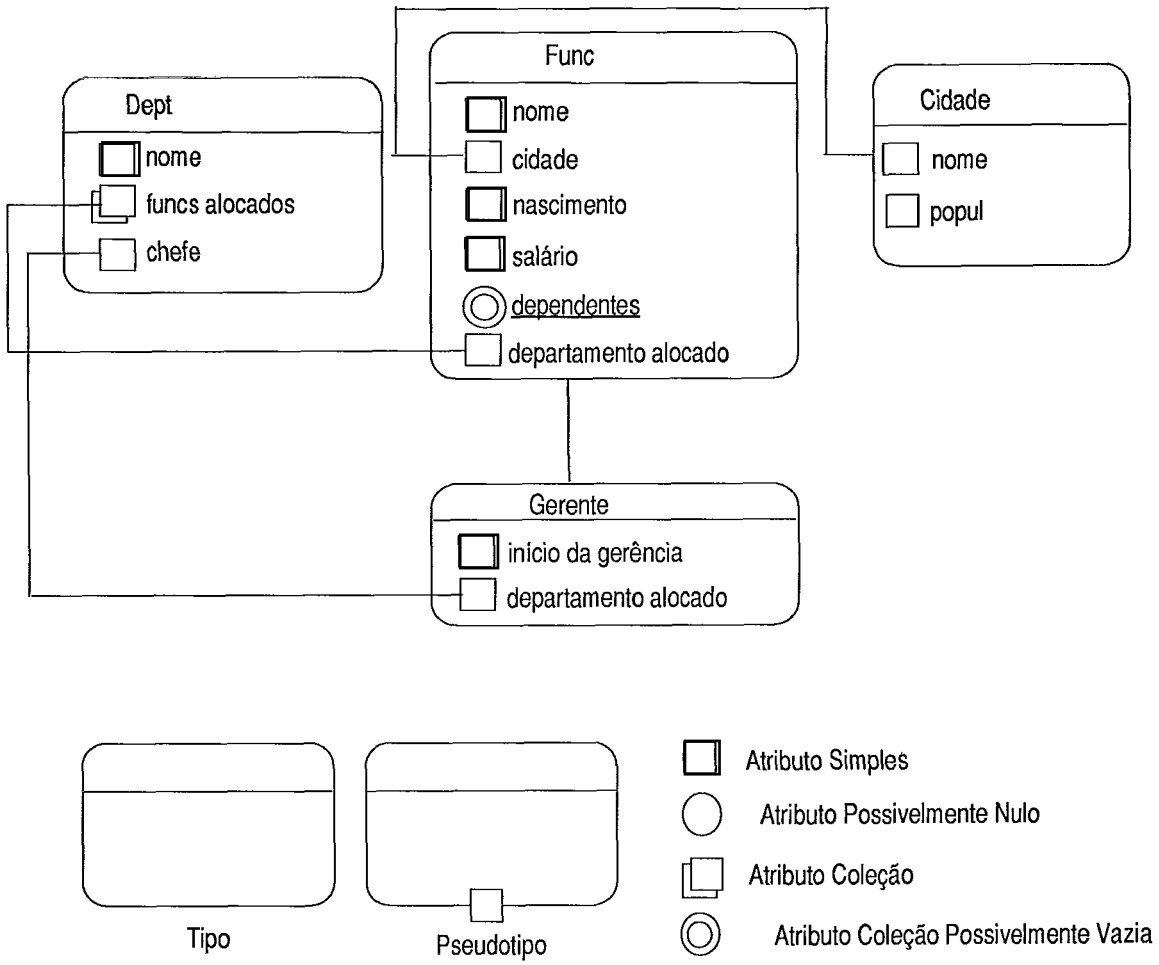

Figura V.3 - Modelagem da aplicação utilizada nos testes

Sempre que a 'cardinalidade' da classe alvo era maior que a da classe fim, a estratégia A/D executava mais rápido que a estratégia descendente(D). Dentre os testes realizados, foram obtidos tempos de execução onde a estratégia A/D gastava apenas 60% do tempo de execução obtido para a estratégia descendente da mesma consulta. Em média, nos testes realizados, a cardinalidade da coleção alvo era de 10 a 30 vezes maior que a cardinalidade da classe do último nível e os tempos de execução da estratégia A/D gastavam de 75% a 60% do tempo da estratégia descendente.

Por outro lado, foi realizado um teste onde a cardinalidade da coleção alvo era 10 vezes menor que a cardinalidade da classe fim e ainda assim foi obtido o mesmo tempo de execução para as duas estratégias (A/D e D). Esse fato pode ser explicado não pela diminuição dos acessos

a disco mas pela ocupação mais eficiente do buffer de páginas, uma vez que inicialmente, o buffer ficou ocupado com os objetos da classe do último nível e posteriormente, a lista dos IDOs ficou na memória e o restante do buffer pode ser ocupado somente com os objetos da classe alvo. Isto é, o algoritmo não teve que dividir o espaço do buffer com os objetos da classe alvo e da classe do último nível.

Finalmente, os resultados preliminares, obtidos sugerem que a estratégia ascendente/descendente pode corresponder a um processamento de consultas mais eficiente que uma implementação puramente descendente. Especificamente no GOA, esta estratégia torna-se ainda mais atraente devido à excução paralela da avaliação de predicados.

Pretende-se na próxima versão do servidor GOA, efetivar a versão experimental da implementação da estratégia A/D, adotando a solução de utilização de listas dos objetos apenas para aquelas classes tidas como candidatas à consultas. Outra opção seria "obrigar" que todos os objetos pertencentes a uma classe do tipo *coleção* estivessem necessariamente inseridos na coleção, isto é, que a população da classe coincidisse com a população da *coleção* associada à classe. Neste caso, a resolução ascendente estaria limitada às classes do tipo *coleção*.

Capítulo VI

O SERVIDOR PARALELO DO SGO GEOTABA

VI.1 PRELIMINARES

Num SGBD relacional, é possível distinguir três fontes de paralelismo. O paralelismo explorado entre transações de diversos usuários, o paralelismo entre as operações que compõem uma transação e o paralelismo dentro de uma operação. O paralelismo intra-operação é o mais explorado e é atingido através de algoritmos paralelos que tiram partido do particionamento dos dados. Como exemplo, pode ser citado o paralelismo num algoritmo da operação de consulta "seleção". A experiência das máquinas paralelas de banco de dados e dos diversos protótipos apresentados no capítulo II (Tabela II.1) vem mostrando que o modelo relacional é naturalmente paralelizável e que o ganho em desempenho que vem sendo atingido é hoje inquestionável.

No caso do modelo de dados orientado a objetos, o paralelismo não é imediato, pois as características dos dados do modelo relacional diferem da orientação a objetos em diversos aspectos. A grande fonte de paralelismo do modelo relacional se apoia no processamento de conjuntos **uniformes** de dados, através de comandos predefinidos, por exemplo, a linguagem de manipulação de dados.

No modelo de dados orientado a objetos, objetos pertecem a classes mas não necessariamente caracterizam um conjunto. Objetos são manipulados por métodos e por alguns "comandos" existentes na interface de acesso e de manipulação de objetos. Além disso, existe uma preocupação na disposição dos objetos no disco de modo a **agrupá-los** de acordo com o melhor padrão de acesso. Embora uma das técnicas de agrupamento seja juntar fisicamente todos os objetos de uma classe, existem outras técnicas que privilegiam o relacionamento de composição entre os objetos, armazenando junto, objetos de classes diferentes [Arau91].

Segundo DeWitt e Gray [DeWi90b] o agrupamento de objetos é ainda um tema de pesquisa em aberto e a combinação com a distribuição de objetos para permitir o paralelismo torna o problema ainda mais complexo. Entretanto, esses aspectos não inviabilizam a exploração de paralelismo em SGBDOOs, mas apenas mostram que técnicas de paralelismo utilizadas em SGBDs relacionais não possuem aplicabilidade direta com os modelos orientados a objetos. Esses fatores levaram a pesquisas de novas fontes a serem exploradas dentro das arquiteturas de SGBDOOs.

A busca de desempenho através do paralelismo levou as arquiteturas de SGBDOOs a adotarem uma plataforma cliente-servidor, onde parte do código do SGBDOO fica no cliente e parte no servidor. O cliente é executado em um processador e o servidor em outro. Com essa distribuição, obtém-se um paralelismo entre a execução dos códigos do cliente e do servidor. Outra fonte de paralelismo que pode ser explorada nos SGBDOOs está na execução das operações predefinidas, ou seja, operações da linguagem de manipulação de objetos. Como exemplo, podem ser citados os comandos da linguagem de consulta e alguns comandos de manipulação. O paralelismo dessas operações pode ser entre operações de usuários e dentro de operações de um único usuário. Entretanto, torna-se necessária a investigação sobre os conflitos entre agrupamento e particionamento para que o paralelismo possa ser explorado efetivamente no modelo orientado a objetos.

Este capítulo, apresenta técnicas de paralelismo para SGBDOOs e propõe uma solução a ser utilizada para aumentar o desempenho do SGBDOO GEOTABA através da exploração de paralelismo. A solução proposta foi experimentada através da implementação de uma versão paralela do servidor de objetos GOA descrito no capítulo V. Nesse sentido, a Seção VI.2 apresenta um panorama geral sobre fontes de paralelismo na orientação a objetos e suas dificuldades de implementação. Nessa Seção, também são abordadas as soluções encontradas na literatura que, no entanto, não contemplam o modelo de dados orientado a objetos. Na Seção VI.3 são discutidas as características da arquitetura paralela com modelo de memória de disco compartilhado e suas implicações na solução adotada para este trabalho. Já na Seção VI.4, é apresentado o paralelismo no contexto da arquitetura do GEOTABA e, finalmente, a Seção VI.5 apresenta o protótipo do servidor paralelo desenvolvido com suas características e avaliação de desempenho.

VI.2 O PARALELISMO E A ORIENTAÇÃO A OBJETOS

Embora o paralelismo na orientação a objetos apresente uma série de dificuldades, esse paralelismo pode ser utilizado junto às aplicações não convencionais de SGBDOOs que necessitam de desempenho. Surge então a necessidade e motivação para que técnicas de paralelismo sejam estudadas para a construção de SGBDOOs paralelos [DeWi90b].

Segundo Oszu e Valduriez [Oszu91], embora as técnicas estejam começando a serem esboçadas, os SGBDOOs distribuídos ou paralelos serão de extrema importância no futuro próximo pelas três razões reproduzidas a seguir:

i) Assim como ocorreu nos bancos de dados relacionais, é muito provável que os SGBDOOs necessitarão de técnicas de bancos de dados distribuídos para prover a disponibilidade dos dados e aumentar o seu desempenho.

ii) As aplicações que precisam das características dos SGBDOOs, em geral, são altamente distribuídas. Como exemplo, podem ser citadas as grandes aplicações CASE ('Computer Aided Software Engineering') que envolvem tipicamente diversos projetistas em cooperação e várias ferramentas distribuídas.

iii) A capacidade de encapsulamento de informações, faz com que os SGBDOOs sejam candidatos naturais ao suporte dos bancos de dados heterogêneos os quais têm a tendência à distribuição.

A utilização de paralelismo num SGBDOO pode envolver desde propostas onde o paralelismo é explorado em todos os níveis do SGBDOO, através da utilização de uma linguagem de programação paralela e orientada a objetos, por exemplo, até a sua utilização em operações delimitadas dentro do SGBDOO. De todo modo, diversas são as questões a serem estudadas ao optar-se pela construção de um SGBDOO paralelo. Segundo DeWitt [DeWi90b], os pontos mais críticos no paralelismo em SGBDOOs estão na exploração de paralelismo nos métodos do usuário e na conciliação do particionamento, necessário ao paralelismo com o agrupamento adequado à gerência de objetos. Esta Seção analisa os problemas e as oportunidades de paralelismo no contexto de SGBDOOs.

VI.2.1 O Uso de Linguagens de Programação Paralela Orientadas a Objetos

Diversos autores [Yone87, Bers88, Jézé92] apontam o mundo de objetos como sendo altamente propício ao paralelismo e concorrência, devido ao encapsulamento dos objetos. Numa situação ideal, pode-se pensar num ambiente paralelo onde cada objeto teria seu processador associado [Agha86]. Cada vez que um objeto recebe uma mensagem, cabe ao seu processador local processá-la, utilizando sua memória local para trabalhar com os dados que também estão encapsulados.

Entretanto, o mundo de objetos não é tão autônomo assim, uma vez que no mínimo o objeto pertence a uma hierarquia de classes, se relacionando implicitamente, com outros objetos. A dificuldade da gerência desta hierarquia faz com que projetos ambiciosos, do ponto de vista do paralelismo, como a linguagem POOL2 [Amer91, Huls91] não permitam a hierarquia de classes no modelo de objetos da linguagem [Amer89]. Este também é o caso da linguagem STRAND que possui construtores que facilitam a definição de atributos e métodos encapsulados, permitindo assim algumas características da programação orientada a objetos. Entretanto, para que STRAND ofereça a concorrência e o paralelismo no alto nível, são limitadas às características da orientação a objetos suportadas pelo compilador da linguagem.

Tais limitações, fizeram com que o uso da linguagem POOL2 fosse abandonado e o uso da linguagem Strand ficou limitado às suas características de programação paralela apenas. A utilização da linguagem POOL2 [Amer91, Huls91] foi analisada no contexto do Projeto TABA, porém a idéia teve que ser abandonada não só devido às limitações no modelo da orientação a

objetos mas pelo fato do código gerado ser executado somente na máquina POOMA, um hardware experimental desenvolvido pela Philips holandesa e portanto de difícil acesso.

VI.2.2 O Paralelismo nas Operações e Métodos do SGBDOO

Conforme sugerido em [Hard88], além do paralelismo proporcionado pela distribuição dos dados, existem diversas outras fontes a serem exploradas. Como exemplo, pode ser citado o paralelismo na gerência do esquema, onde o gerente pode varrer a hierarquia de classes paralelamente, na busca da definição de um atributo ou método. A linguagem de definição de métodos também pode conter comandos onde o usuário especifica operações a serem executadas em paralelo. Entretanto, um projeto onde todo o SGBDOO é executado sobre uma máquina paralela, pode ser chamado de uma proposta "revolucionária", pois neste caso, todos os gerentes do SGBDOO deverão explorar o paralelismo mais adequado à sua gerência. Considerando que a tecnologia de banco de dados na orientação a objetos não se encontra suficientemente dominada, a proposta deste trabalho concentra-se no paralelismo direcionado às operações do servidor de objetos.

A questão do paralelismo dentro dos métodos escritos pelo usuário ainda é um tema em aberto, pois é dependente da linguagem utilizada pelo usuário. Entretanto, três opções podem ser vislumbradas. Na primeira opção, se o usuário faz uso da linguagem de programação própria do banco de dados ou utiliza os comandos da linguagem de consulta, não há problemas pois o otimizador junto ao executor dos comandos poderá tirar proveito de uma execução paralela de modo transparente ao usuário. Outra opção é fazer com que o usuário programe seus métodos em uma linguagem de programação paralela. Numa terceira opção, o paralelismo do método seria explicitado pelo usuário. Caso seja utilizada uma linguagem de programação convencional, seria necessária a utilização de um pré-processador que identificaria comandos do tipo 'parallel do', por exemplo.

Devido à dificuldade do SGBDOO em tirar proveito da otimização dos métodos escritos pelo usuário, por conta de sua natureza procedimental, diversos autores [Ston90, Oszu91, Kim 90a] sugerem que o usuário deva ser incentivado ao máximo para a utilização de operadores predefinidos, a exemplo dos comandos da linguagem de consulta na codificação de seus métodos. Desta forma, poderão ser aplicadas técnicas de otimização, objetivando que e o usuário consiga mais independência de dados, facilitando os mecanismos de evolução do esquema, por exemplo. Do mesmo modo, num SGBDOO com paralelismo, métodos que utilizem os comandos predefinidos poderão se beneficiar do conhecimento que o otimizador tem destes comandos, no que concerne à paralelização.

Para tentar contornar o problema de paralelização dos métodos, a solução adotada foi a de limitar o escopo do paralelismo do GEOTABA ao servidor de objetos GOA. Como as operações executadas no GOA são predefinidas, o paralelismo pode ser explorado sem muitos

problemas. Desta forma, o usuário irá tirar proveito do desempenho do sistema através de um paralelismo transparente, principalmente quando faz uso dos operadores da linguagem de consulta.

VI.2.3 A Questão do Agrupamento X Particionamento

Como na orientação a objetos o conceito de classes não coincide com o de conjuntos, ao contrário do modelo relacional, é necessário que se escolha sobre que grupo de objetos será realizado o particionamento dos dados. Segundo DeWitt [DeWi90b], é preciso optar entre um particionamento de objetos aplicado a todas as classes e entre um particionamento onde somente classes do tipo *coleção* possuem seus objetos distribuídos. Além disso, deve ser considerado um particionamento sobre coleções de objetos que são referenciadas em atributos multi-valorados.

Uma vez identificadas as classes ou coleções que terão seus objetos distribuídos através de alguma técnica de fragmentação, surge a questão sobre como conciliar a indicação de **particionamento** de uma coleção para o processamento paralelo, com a indicação conflitante de **agrupamento** para o processamento em conjunto. Outro problema que surge neste conflito está em que fazer quando os objetos referenciados são armazenados junto ao objeto raiz da hierarquia de composição, ao mesmo tempo que os objetos referenciados pertencem a uma classe do tipo coleção.

Para tentar resolver a questão da dicotomia entre o particionamento e o agrupamento de objetos, foi adotada a solução conciliatória da arquitetura paralela de disco compartilhado. Nesta arquitetura, ao contrário da memória distribuída, os dados/objetos não precisam estar fisicamente distribuídos e alocados aos processadores. Cabe ao Gerente de Distribuição escolher a melhor maneira de "enviar" os objetos aos processadores. Desta forma, o processamento seqüencial mantém o mesmo padrão de acesso aos objetos em disco e o processamento paralelo realiza a distribuição dinamicamente no momento da execução da operação paralelizável.

VI.2.4 As Soluções Anteriores

Poucas são as propostas encontradas na literatura para uso de paralelismo em SGBDOOs. Existem projetos que apresentam soluções no âmbito dos chamados bancos de dados de terceira geração [Ston90] que possuem a proposta de suportar aplicações não convencionais através de extensões ao modelo relacional. Como exemplo, podem ser citados os projetos/protótipos EDS [Vald90], PRIMA [Mits92] e XPRS [Hong92].

Nestes sistemas, a aplicação pode ser modelada com estruturas mais ricas que o modelo relacional, permitindo uma modelagem com semântica através de objetos complexos entre outras características. Entretanto, o modelo interno adotado ainda é baseado em estruturas e operações

da álgebra relacional. Como o paralelismo nestes sistemas é empregado apenas no nível interno do processamento das operações, as técnicas de paralelismo utilizadas nos SBDPs relacionais são quase que diretamente aplicáveis.

O projeto EDS, conforme apresentado no capítulo II, utiliza uma extensão da linguagem SQL (ESQL) para a manipulação dos objetos que é mapeada para uma extensão da álgebra relacional (LERA). É no contexto da linguagem LERA que são realizadas as operações paralelas. A arquitetura do sistema EDS também adota a solução de cliente/servidor com o servidor paralelo de dados com capacidade de processar operações da álgebra relacional.

No projeto XPRS, também é realizado o paralelismo nas operações da álgebra e pretende-se que o processamento paralelo seja integrado ao sistema PostGres [Ston90b]. Já no projeto PRIMA, é utilizado um modelo de dados próprio, o MAD [Mits91] que também é mapeado para extensões da álgebra relacional. A arquitetura do PRIMA, utiliza a divisão cliente e servidor de objetos, onde o servidor de objetos deverá ser executado sobre uma máquina paralela de memória compartilhada da Sequent.

Outra utilização do paralelismo no contexto da orientação a objetos reside no projeto VOLCANO [Grae88]. Entretanto, os resultados práticos disponíveis na literatura, são encontrados apenas no contexto do processamento de consultas do modelo relacional [Grae90] com algumas características de objetos complexos. A idéia do sistema VOLCANO está no desenvolvimento de um processador paralelo de consultas extensível. A característica da extensibilidade permite que novas operações de consulta sejam facilmente incorporadas. Além disso, o sistema foi projetado de modo a ser passível de utilização por diversos gerenciadores de banco de dados. O sistema VOLCANO é executado sobre uma máquina paralela de memória compartilhada, também da Sequent, e gerencia a execução paralela das operações através de uma técnica denominada pelos autores de *modelo de operadores*, onde são usados processos 'produtores/consumidores' para a avaliação de consultas.

Assim como na proposta do servidor paralelo do GOA, estes projetos limitam o paralelismo às operações realizadas no contexto do servidor de dados e especificamente no processamento de consultas. Por outro lado, pode ser observado que a maioria das soluções existentes utiliza uma proposta conservadora, fazendo uso do paralelismo num servidor de dados relacional que se comunica com clientes, onde os dados são mapeados para um modelo mais rico em semântica.

Valduriez e Öszu [Öszu91] apresentam uma discussão sobre técnicas a serem utilizadas no armazenamento distribuído de objetos e no processamento de consultas sobre os objetos distribuídos. Entretanto, as soluções apresentadas sempre recaem no mapeamento da orientação a objetos para o modelo relacional na representação física dos objetos. Conforme discutido no capítulo IV, esse mapeamento pressupõe uma associação de classes com relações, ou seja, o sistema teria que manter a extensão (população de objetos) de cada classe definida no esquema o

que implica em uma série de problemas como redundância, entre outros já citados na Seção V.4. Sendo assim, a solução do servidor paralelo do GOA adota um modelo interno de representação de **objetos** e implementa versões paralelas para os algoritmos de processamento de consultas no contexto da orientação a objetos.

VI.3 A SOLUÇÃO ADOTADA NA GERÊNCIA PARALELA DE OBJETOS

Para solucionar os problemas de paralelismo na orientação a objetos apresentados na Seção VI.2, foram adotadas as seguintes estratégias:

i) Optou-se pela utilização da arquitetura de disco compartilhado (DC) no sentido de conciliar os requisitos de agrupamento com o particionamento.

ii) Para atacar o problema de paralelismo sobre métodos, foi limitado o escopo das operações paralelizáveis através da utilização do servidor paralelo de dados [Vald90a] que privilegia as operações sobre conjuntos de objetos. Nesse sentido, os métodos poderão usufruir de paralelismo quando utilizarem as operações predefinidas do servidor de objetos, como por exemplo, as operações de consulta.

A seguir serão consideradas características da arquitetura de disco compartilhado e sua adequação à ortogonalidade existente entre agrupamento e distribuição.

VI.3.1 Características da Arquitetura de Disco Compartilhado

Na arquitetura de disco compartilhado (DC), os processadores têm acesso a todos os discos, mas cada processador tem sua memória própria e privativa. Desta forma, uma das vantagens da arquitetura DC é que os dados não precisam estar distribuídos fisicamente através de discos de acesso local aos processadores. A distribuição dos objetos pode ser feita no momento do acesso aos dados. Desta forma, os objetos possuem uma disposição no disco não comprometida com o acesso paralelo às coleções, mas sim com a melhor estratégia de acesso à base de um modo global.

Pelo fato de todos os processadores terem acesso às páginas do disco, torna-se necessário que as operações de leitura e gravação sejam controladas. Para evitar a inconsistência entre os 'buffers' e o conflito entre bloqueios, são apontadas por Bhide [Bhid88b] quatro técnicas para controle do acesso concorrente às páginas do disco compartilhado:

a) DIS - 'Disk Controller Locking' - Esquema distribuído.

O próprio software que controla o disco mantém uma tabela de bloqueios e quando é feito o pedido da página, a solicitação de bloqueio já vai junto. Isto é, existem comandos para o DIS que solicitam bloqueio.

b) CLM - 'Central Lock Manager' - Esquema centralizado.

Um dos processadores é alocado para fazer a gerência do controle do bloqueio. Os outros processadores solicitam o bloqueio e liberação para o acesso a determinadas páginas ao gerente dos bloqueios.

c) PRI - 'Primary Copy Method' - Esquema distribuído.

A base de dados é dividida em pedaços e cada pedaço é alocado a um processador. Os processadores trocam mensagens sincronamente solicitando os bloqueios.

c) APRI - 'Asynchronous Primary Copy Method' - Esquema distribuído.

A idéia é a mesma do PRI só que ganha tempo em sendo assíncrono. Baseia-se no esquema otimista de concorrência, já que o processador que solicita o bloqueio, não espera pela resposta e começa imediatamente a trabalhar com as páginas requisitadas. Quando chega a resposta da solicitação o processador a analisa e caso o pedido tenha sido negado, desfaz a transação.

Um dos problemas da arquitetura DC consiste na validação de 'buffers' já que a mesma página poderá estar em diversos processadores ao mesmo tempo. Para evitar a inconsistência entre os 'buffers' são apresentadas várias técnicas em [Bhid88b]. Entretanto, Bhide sugere a utilização da técnica de liberação de 'buffers' ('buffer purge'), isto é, quando a transação acaba todos os 'buffers' modificados são escritos no disco. Com isso, não existe 'buffer hit', ou seja, a página necessitada já estar em memória, mas simplifica a implementação. Assume-se que a área de 'buffers' é grande o suficiente para manter todas as páginas alteradas da transação ao mesmo tempo. Na implementação realizada, esse esquema foi adotado com algumas modificações comentadas na Seção VI.5.3.

VI.3.2 Comparando os Diversos Modelos de Memória

Não existe um consenso quanto ao melhor modelo de memória a ser adotado em uma arquitetura de banco de dados, e diversos trabalhos [Bhid88a,b, Laks89, DeWi90c, DeWi92] dividem-se favoravelmente a um dos três modelos apresentados no capítulo II. Bhide e Stonebraker apresentam um estudo comparativo de desempenho [Bhid88a] com as arquiteturas de memória totalmente compartilhada (MC) e memória totalmente distribuída (MD).

Em [Bhid88b] a arquitetura DC é acrescentada por Bhide ao estudo comparativo já divulgado. Nesse trabalho, Bhide apresenta a arquitetura MC como a de melhor desempenho, seguida da arquitetura DC e com o pior desempenho o modelo MD. Além disso, Bhide também sugere a utilização da arquitetura DC para casos em que não seja possível o particionamento físico da base de dados. Os trabalhos de Lakshimi e outros [Laks89,90] também apontam o modelo de memória MD como sendo o de pior desempenho.

Entretanto, deve-se observar as limitações do hardware da arquitetutra de MC, a saber, a largura de banda da memória que limita o número de processadores a serem utilizados. Atualmente, as arquitetutras MC não possuem mais do que 32 processadores acessando a mesma memória. Nesse sentido, diversos SBDPs relacionais apresentados no capítulo II, Gamma, Bubba, DBC e NonStopSQL apostam na expansibilidade da arquitetura com MD para superar o desempenho.

Segundo DeWitt e Gray [DeWi92] a possibilidade de crescimento quanto ao número de processadores e discos da arquitetura DC, esbarra na dificuldade do controle da concorrência. Neste trabalho, os autores apontam que a arquitetura DC não é muito eficiente para aplicações que realizam muitas leituras e gravações numa base de dados compartilhada, uma vez que são grandes as chances de inconsistência entre os 'buffers' dos processadores, o que acaba implicando em diversas trocas de mensagens para realizar o controle, vide as técnicas de PRI e APRI comentadas na Seção anterior. Sendo assim, DeWitt e Gray sugerem que as soluções a serem adotadas no controle de concorrência do DC tenderão ao modelo da memória distribuída.

Devido à grande discussão em torno do modelo de memória a ser adotado, Amorim [Amor92] considera que o modelo de memória mais adequado é muito dependente da aplicação e nesse sentido, no projeto da próxima versão do NCP os modelos MD e DC conviverão no nível do hardware e a escolha do modelo poderá ser feita pelo projetista do software da aplicação.

Apesar das vantagens e desvantagens apresentadas quanto ao modelo de memória de DC, este foi o modelo adotado por ser o mais adequado aos requisitos da orientação a objetos e à proposta de paralelismo deste trabalho para a arquitetura do GEOTABA. A idéia inicial do servidor paralelo de objetos não é trabalhar com um grande número de processadores, mas de analisar a viabilidade da utilização de paralelismo junto à gerência de objetos para contribuir para o desempenho do SGBDOO.

VI.4 O PARALELISMO NA ARQUITETURA DO GEOTABA

Devido aos diversos problemas encontrados no particionamento dos dados orientados a objetos e principalmente devido à dificuldade de execução de métodos em paralelo, surgiu a opção do servidor paralelo de objetos. O servidor limita as operações a serem paralelizadas, porém engloba o processamento de consultas que ainda é uma das maiores fontes de paralelismo.

Uma proposta revolucionária para o desenvolvimento de um SGBDOO paralelo poderia envolver aspectos como a utilização de uma linguagem paralela orientada a objetos, ou a utilização de algoritmos paralelos em todos os níveis da arquitetura do SGBDOO. Entretanto, o uso da orientação a objetos em SGBDs ainda não atingiu maturidade científica, e a incorporação de uma máquina paralela para hospedar todo o SGBDOO implicaria numa complexidade difícil de

ser gerenciada. Desta forma, optou-se por uma solução onde o paralelismo fica direcionado às operações do servidor de objetos, com ênfase na manipulação de coleções de objetos.

A proposta de paralelismo na arquitetura do GEOTABA consiste, então, da substituição do servidor GOA instalado sobre uma estação de trabalho seqüencial, por um servidor paralelo GOA instalado sobre uma máquina paralela, o NCP I. A Figura VI.1 mantém a mesma divisão cliente/servidor entre os gerentes do GEOTABA da Figura IV.2, apenas mudando o protocolo de comunicação e parte do código dos módulos do servidor. O protocolo de comunicação não pode continuar usando a ferramenta 'rpc', entretanto, como o sistema operacional do NCP I, o Helios é uma versão paralela do Unix, o protocolo a ser utilizado não oferece muitos problemas e será adotado o TCP/IP [Sinh92]. Maiores detalhes são apresentados no relatório [Matt93b].

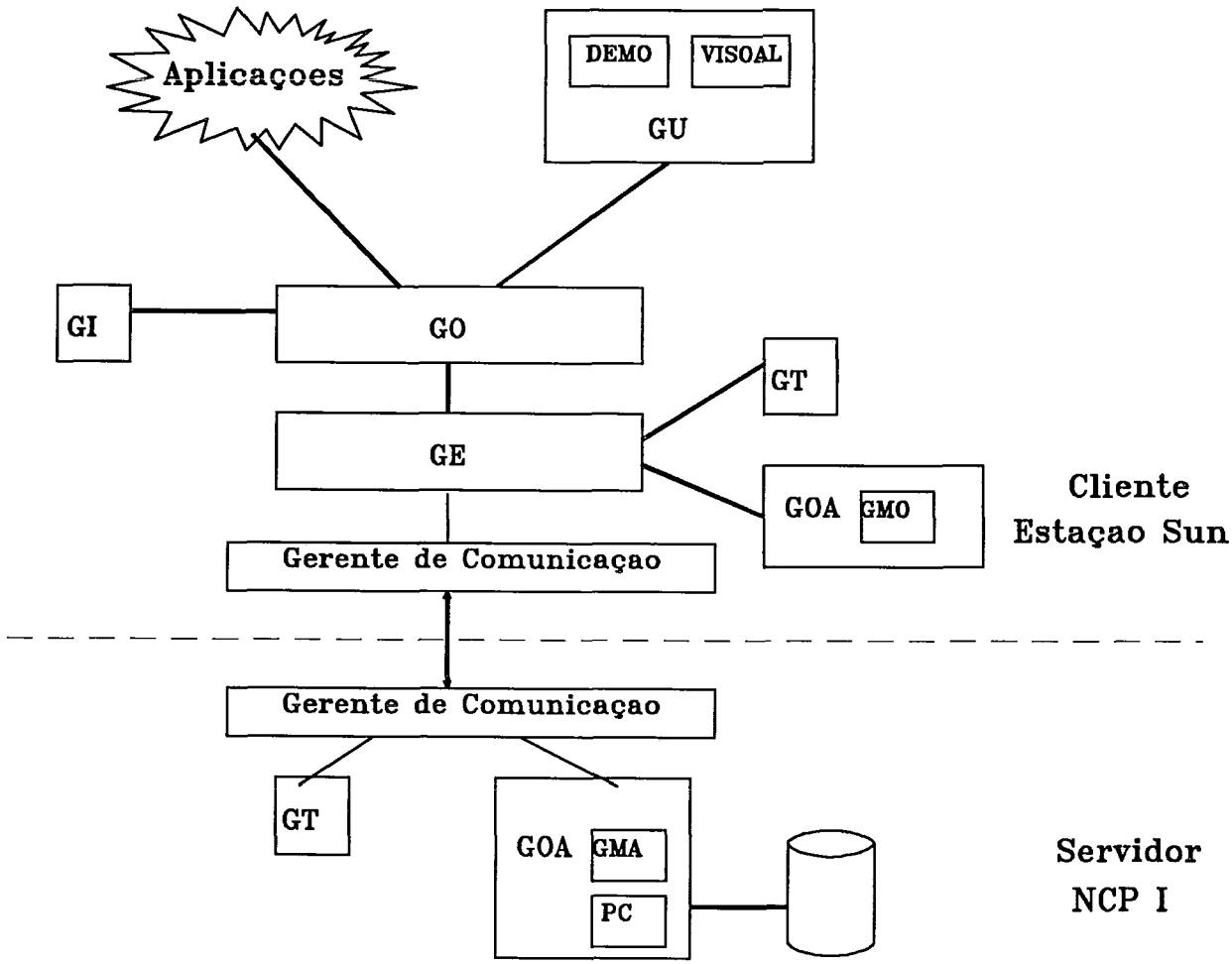

Figura VI.1 - Arquitetura cliente/servidor paralelo do GEOTABA

As operações que o cliente solicita ao servidor paralelo GOA são exatamente as mesmas apresentadas na Seção V.4 do servidor seqüencial. Entretanto, antes da execução, o Gerente de Distribuição analisa que nível de granularidade pode ser aplicado à operação. Caso a operação

possa ser distribuída entre os diversos processadores, o paralelismo é *dentro da operação* e a operação é executada em paralelo. Caso a operação não seja intra-paralelizável, como por exemplo, a solicitação do cliente para que lhe seja enviado um determinado objeto, o Gerente de Distribuição aloca um único processador para executá-la e verifica se outras operações podem ser executadas nos demais processadores. Ao término da execução da operação, o Gerente de Distribuição envia o resultado para o cliente.

Cabe observar que, devido aos diversos problemas existentes entre o paralelismo e a orientação a objetos, a solução adotada preocupa-se com a conciliação de casos que não são naturalmente paralelizáveis. Nestes casos, o seu processamento é realizado em um único processador deixando os demais processadores livres para a execução de outras operações. Espera-se que, no pior caso, a solução do servidor paralelo obtenha o mesmo desempenho do servidor seqüencial. Isto porque, não foram realizadas modificações na gerência de armazenamento dos objetos e nestes casos, os processadores não terão a sobrecarga da comunicação.

É importante notar também que o uso do servidor de objetos permite que várias fontes de paralelismo sejam exploradas, principalmente através do processamento de consultas. Segundo Stonebraker [Ston90], à medida que são utilizadas transferências de dados em níveis mais baixos, do tipo páginas ou registros, o protocolo utilizado também é mais baixo e consequentemente mais complexo. Stonebraker cita também, no âmbito dos sistemas relacionais, experiências [Tand88] que mostram a superioridade do desempenho ao se utilizar uma comunicação cliente/servidor via SQL em detrimento a interfaces de nível mais baixo. Imagina-se que o mesmo pode ser concluído no que concerne o servidor de objetos com relação aos servidores de páginas e arquivos.

Além disso, estudos de DeWitt e outros [DeWi90c] demonstram que a opção do servidor de objetos é a menos sensível ao agrupamento de objetos no disco, já que a memória do cliente só contém os dados necessários ao cliente. Essa característica reforça ainda mais a opção pelo servidor de objetos paralelo.

Como desvantagem, o servidor de objetos limita suas fontes de paralelismo na medida em que informações do esquema são gerenciadas pelo cliente somente. O compartilhamento de informações do esquema entre cliente e servidor implicaria em um aumento considerável da complexidade do servidor, principalmente por conta do controle da redundância. Nesse sentido, esta versão do PARGOA fica limitada às informações do esquema do modelo interno de representação, disponíveis no servidor.

VI.5 O PROTÓTIPO PARGOA

PARGOA é um protótipo do servidor paralelo de objetos do GEOTABA. A implementação deste protótipo une as experiências obtidas a partir do desenvolvimento do

PARBASE e do servidor de objetos GOA. O resultado foi uma mistura de módulos dos protótipos já desenvolvidos, onde parte do código do PARBASE foi reutilizado para o Gerente de Distribuição do PARGOA e as operações seqüenciais dos nós consistem de pequenas adaptações ao código do servidor do GOA.

Além das vantagens da utilização do mesmo ambiente do PARBASE, citadas na Seção II.3, versões mais recentes do NCP I contam com uma série de atrativos extras descritos a seguir.

VI.5.1 O Ambiente de Desenvolvimento

O **ambiente de desenvolvimento** do PARGOA foi o mesmo utilizado para a implementação do PARBASE, descrito no capítulo III. Devido a testes ainda em realização sobre o computador paralelo NCP I, que conta em cada nó com um transputer e um microprocessador i860 (Figura III.1), a implementação do PARGOA também foi limitada ao modelo T8 do NCP I, onde cada nó contém um transputer e 2 Mbytes de memória. O computador hospedeiro utilizado foi um PC-AT, entretanto, o disco rígido utilizado foi de capacidade inferior ao modelo T8 das experiências do PARBASE. Sendo a unidade de disco rígido um ponto crítico para um SGBD, os resultados obtidos ficaram um pouco prejudicados. Entretanto, no protótipo do NCP I em experimentação e não aberto ainda aos usuários, já está instalado, dentro do modelo de memória de disco compartilhado, um disco rígido com capacidade de armazenamento de 676 Mbytes de informação, acessado pelos 8 processadores através da interface SCSI - 'Small Computer System Interface' que viabiliza o acesso concorrente. Esse disco é gerenciado pelo sistema operacional Helios, possui um 'cache' próprio e num futuro bem próximo a implementação do PARGOA irá tirar proveito desta nova configuração. Os testes realizados com o disco e a nova interface SCSI apontaram um ganho na velocidade de transmissão de até 40 vezes mais rápido em relação à utilização do disco rígido com controladora MFM, atualmente no PC. Além disso, os nós já contam com uma memória local com 4 Mbytes de armazenamento (o dobro da capacidade atual e utilizada nos testes).

Embora tenha sido utilizado um PC como computador hospedeiro, o NCP I está sendo integrado com a rede de Sun's do Programa de Engenharia de Sistemas e Computação da COPPE, onde uma estação Sun servirá de hospedeira do NCP I colocando em prática a arquitetura proposta para o GEOTABA da Figura VI.1.

Quanto ao **ambiente de programação**, foi utilizada a linguagem Strand para as rotinas de gerência do paralelismo e foi utilizada a linguagem C para o código seqüencial carregado nos processadores. Na realidade, o código carregado nos nós é exatamente igual ao código utilizado pelo servidor seqüencial do GOA nas estações Sun. O código foi totalmente compatível com o C do sistema operacional Helios do NCP I. Conforme esperado, não foi necessária nenhuma modificação para o transporte do código. A característica do STRAND de permitir a associação direta de código a processadores, aliado à sua interface para a linguagem C, fizeram com que

qualquer operação do servidor seqüencial do GOA pudesse ser executada por qualquer processador do NCP I.

VI.5.2 Características do PARGOA

Apesar do PARGOA oferecer diversas oportunidades de paralelismo em sua execução, algumas limitações foram realizadas na implementação corrente, pois o objetivo principal do protótipo foi de analisar o grau de dificuldade ou facilidade na exploração do paralelismo. Foi realizado, antes de mais nada, o estudo de viabilidade da utilização de processamento paralelo no contexto da gerência de dados com orientação a objetos. A seguir são apresentadas as opções realizadas quanto às técnicas de paralelismo escolhidas no projeto do PARGOA.

i) Nível de granularidade

No nível de granularidade do processamento paralelo de operações, o PARGOA permite que os três níveis apresentados no capítulo II sejam explorados. O paralelismo dentro de operações é explorado por meio das operações de avaliação de predicados do servidor através da distribuição dos objetos da coleção alvo. Já no nível entre operações, pode ser citado a realização do armazenamento paralelo de dois objetos de classes diferentes, ou da modificação de um objeto e da remoção de outro, entre outras operações.

Embora o paralelismo entre transações, possa ser explorado, as aplicações dos SGBDOOs em geral possuem transações longas e a gerência desse paralelismo torna-se mais difícil. Além disso, a transação longa pode envolver diversas operações gráficas de interface com o usuário e a execução de métodos programados pelo usuário. Tais características fazem com que se desconheça as operações sendo realizadas, para que o paralelismo possa ser explorado, a menos que seja explicitado pelo usuário ou que utilize operações predefinidas.

ii) Modelo de memória

Conforme já exposto na Seção VI.3, a implementação do PARGOA tira proveito da arquitetura de disco compartilhado do NCP I que não impõe uma distribuição física dos objetos pelos processadores. Cada processador dispõe de memória local para o 'buffer' de páginas e tem acesso ao mesmo espaço em disco. O modelo DC se mostrou bastante adequado ao esquema de armazenamento de objetos.

iii) Fragmentação das coleções

Devido aos problemas apresentados quanto à dificuldade de conciliar o agrupamento da orientação a objetos com o particionamento do processamento paralelo, várias estratégias foram avaliadas. O ponto de partida da implementação foi não prejudicar as técnicas de agrupamento adotadas no servidor de objetos seqüencial do GOA. Para escolher a técnica de

distribuição, as características do modelo interno e agrupamento dos objetos tiveram que ser consideradas. Como por exemplo:

(a) O escopo da coleção alvo, contém somente os objetos inseridos na coleção e não necessariamente todos os objetos da classe associada a coleção alvo.

(b) O acesso a extensão de uma classe não está disponível, nem mesmo a informação de que essa classe seja do tipo coleção.

- Os objetos de classes do tipo coleção são preferencialmente armazenados em segmentos de tamanho fixo exclusivos para o armazenamento da coleção. À medida que um segmento fica cheio, outros vão sendo criados.
- Já objetos complexos compostos podem ter indicação para serem armazenados junto com seus sub-objetos, mesmo que estes pertençam a uma coleção.

(c) os objetos de uma coleção estarão quase sempre armazenados contiguamente.

Como não existe um comportamento/armazenamento uniforme entre os objetos das classes definidas na base, a estratégia de distribuição dos objetos e alocação dos processadores tenta conciliar dois objetivos, a saber: 1) minimizar o número de acessos a disco tirando proveito da estratégia de armazenamento, e 2) minimizar o acesso concorrente a páginas compartilhadas no disco.

Nesse sentido, a fragmentação só ocorre no processamento de coleções. As coleções são distribuídas através da fragmentação **horizontal,** por analogia ao modelo relacional, onde **objetos** (tuplas), e não atributos, são distribuídos pelos processadores. Diferentemente do PARBASE, o desagrupamento no PARGOA só é total quando a coleção utiliza **n** ou mais segmentos, onde **n** equivale ao número de processadores da configuração da máquina paralela. Foi adotado o **desagrupamento parcial**, uma vez que os resultados obtidos com o PARBASE e trabalhos analíticos [Bora88], apontam diversas vantagens sobre o desagrupamento total, que no entanto é mais fácil de ser implementado.

iv) Política de Distribuição

A política de distribuição de objetos de uma coleção é uma variação da técnica circular ('round robin'). No momento da execução de uma operação de consulta, os segmentos que armazenam fisicamente os objetos de uma coleção são distribuídos entre os processadores, sem levar em conta os valores dos objetos. Foram avaliadas as opções de utilização de uma distribuição por faixa de valores, por exemplo, através do uso de índices parciais [Ston88]. Entretanto, a limitação das informações semânticas conhecidas pelo servidor de objetos, torna complexa a manutenção desses índices. Por exemplo, quando um objeto é modificado no cliente, o servidor não possui a informação sobre que atributos foram alterados. Esse tipo de estratégia de distribuição será avaliado em versões futuras, uma vez que contribuem para o desempenho do SGBD através da localidade conhecida, seja para a faixa de valores ou seja

pela função de 'hashing' utilizada. Essa contribuição pode ser comprovada através dos experimentos do PARBASE.

VI.5.3 A Implementação do PARGOA

Para implementar todas as características do PARGOA (descritas na Seção anterior), foram definidos os módulos apresentados na Figura VI.2. Esses módulos tiram proveito das implementações do PARBASE e do GOA seqüencial.

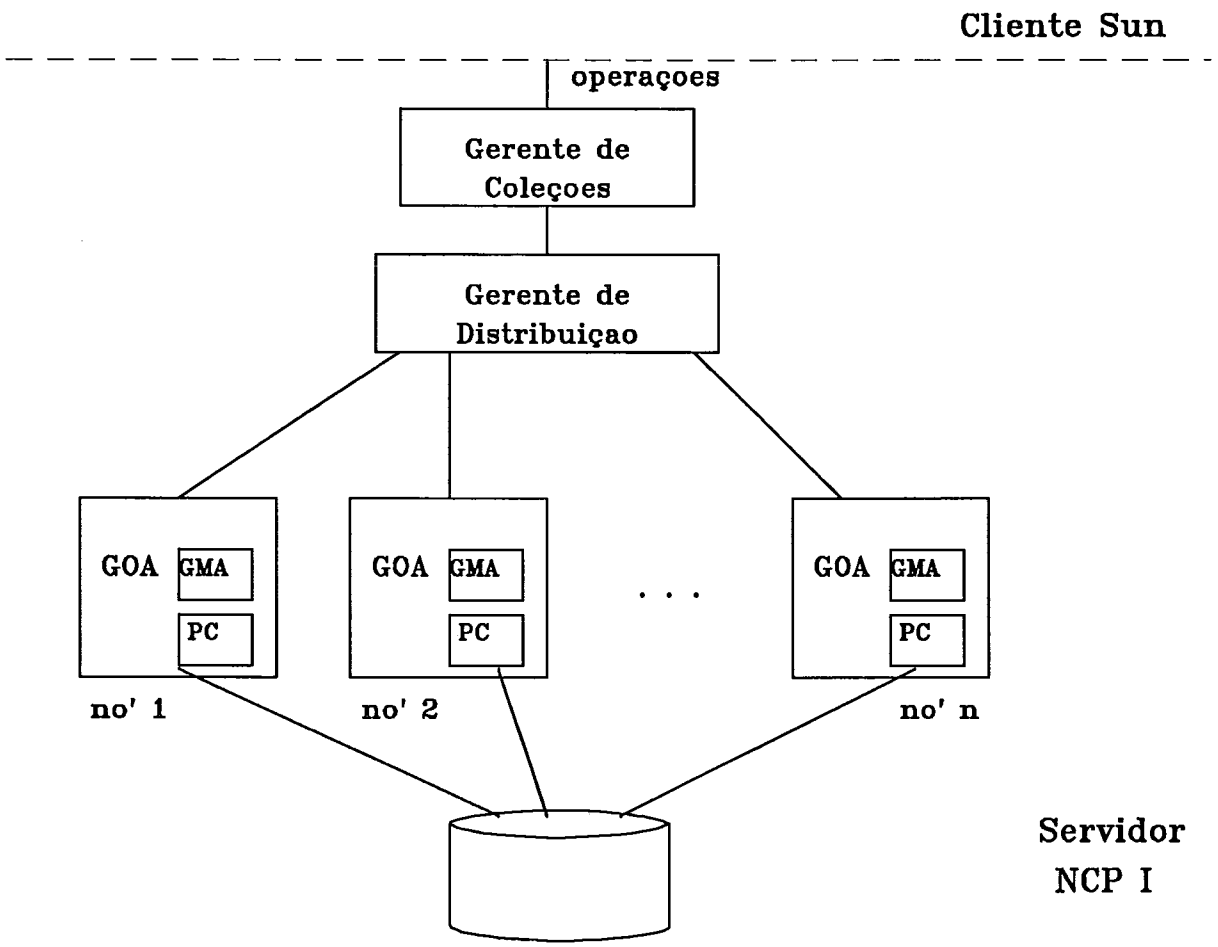

Figura VI.2 - Módulos do servidor paralelo de objetos do PARGOA

Assim como no servidor seqüencial do GOA, o cliente possui à sua disposição todas as operações descritas na Seção V.4. Embora o protocolo de comunicação tenha mudado, o cliente requisita a execução de uma operação com o mesmo código intermediário especificado para o servidor seqüencial.

Ao receber a operação codificada, o servidor envia para o **Gerente de Coleções**, responsável pela obtenção de algumas informações descritivas da coleção ou classe envolvida.

O **Gerente de Distribuição** é o responsável pela análise das características da operação para verificar se ela será executada em **1** ou **n** processadores e se é necessário a utilização de mecanismos de sincronização entre uma operação e outra, ou dentro da mesma operação. No caso do processamento de uma avaliação de predicado, o Gerente de Distribuição analisa a cardinalidade da coleção e o número de segmentos envolvidos. É enviado para cada processador, o código da operação e uma lista dos IDOs, local ao processador.

A implementação atual do PARGOA, não conta ainda com um controle de concorrência para garantir a consistência entre as páginas dos 'buffers' locais aos processadores. Desta forma, as operações paralelizáveis no PARGOA estão limitadas às operações que não envolvem gravações no disco, como é o caso do processamento de consultas do GOA. Pretende-se no futuro que o Gerente de Distribuição entre em sintonia com o Gerente de Transações para que seja adotada a técnica 'CLI' para implementar o controle centralizado para o acesso concorrente às páginas do disco.

O **Gerente de Objetos Armazenados** de cada processador é quase igual ao GOA seqüencial, já que nos nós o processamento é serial. A operação é identificada e o controle é desviado para o algoritmo responsável pela sua execução. Cada processador possui seu 'buffer' de páginas que é gerenciado com as mesmas rotinas do GOA seqüencial, entretanto, ao término da operação, os 'buffers' são liberados. Existe, no entanto, um tratamento de excessão para o buffer do nó 0, uma vez que o Gerente de Distribuição tem total controle sobre essa área. Neste caso, o buffer do nó 0 é gerenciado do mesmo modo que no GOA seqüencial e faz o papel do buffer de páginas dos clientes.

Foi dado um destaque ao módulo responsável pelo processamento de consultas no servidor paralelo, o PC, pois a avaliação de cláusulas de predicados no GOA paralelo é uma das grandes fontes de paralelismo explorada. O código seqüencial que fica carregado nos nós do NCP I, para realizar as operações de consulta, difere dos algoritmos do GOA seqüencial, na medida em que os processadores já recebem a lista de IDOs a ser percorrida e não a identificação da coleção como um todo. De modo análogo, as demais operações do GOA paralelo possuem pequenas variações em relação ao código seqüencial.

Embora a utilização do Gerente de Distribuição imponha uma centralização na gerência do paralelismo, ela é minimizada pelo modelo de disco compartilhado e pela topologia hipercúbica do NCP I. No modelo de disco compartilhado, uma vez tendo sido distribuídas as operações pelos nós, cada processador realiza sua comunicação com o disco em paralelo, sem envolver o Gerente de Distribuição. Na topologia do NCP I (Figura VI.3), o nó 0 intermedia a comunicação com o computador hospedeiro. Desta forma, o Gerente de Distribuição fica carregado no nó 0, por onde o código de chegada e saída do cliente teria que passar de qualquer maneira.

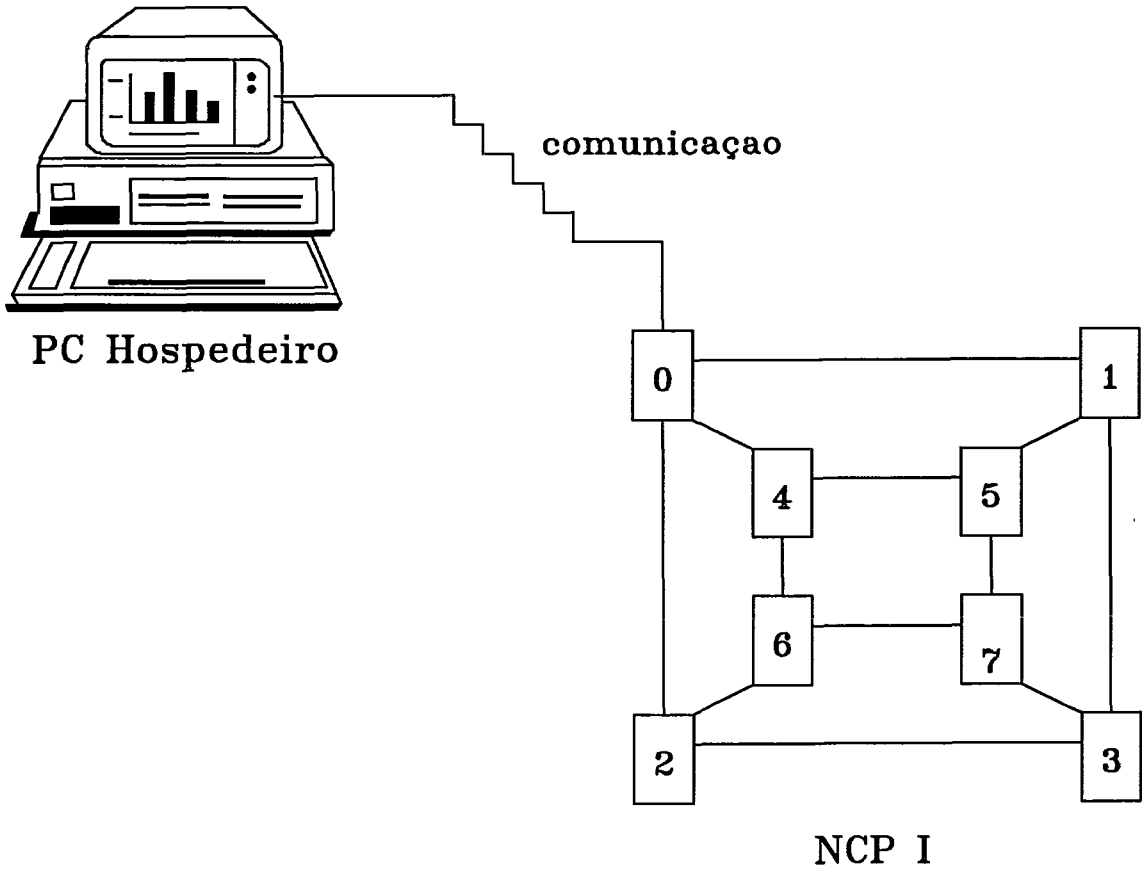

Figura VI.3 - Comunicação NCP I com hospedeiro

VI.5.4 Análise das Operações de Consulta

O objetivo desta análise não é ser extensa ou completa através da medida de desempenho das operações de consulta, mas sim de mostrar a viabilidade do processamento paralelo em operações que utilizam um modelo de dados orientado a objetos.

Foram realizados testes com avaliações de predicados através da estratégia descendente 'nested loops'. O Gerente de Distribuição analisa o número de objetos da coleção e opta por uma distribuição total ou parcial.

Na distribuição total, são distribuídos os segmentos que contém os objetos da coleção pelos processadores. Os objetos que não estão armazenados nos segmentos da coleção, são distribuídos de acordo com seu endereço físico. Os endereços são ordenados e divididos pelo número de processadores. Essa distribuição visa a obtenção de um bom balanceamento de carga dos objetos da coleção alvo entre os processadores, além de minimizar o acesso concorrente a páginas compartilhadas pelos processadores através de uso dos endereços físicos. Cada

processador percorre sua lista de objetos penetrando nos níveis do acesso aos objetos das diversas classes envolvidas no caminho do predicado.

Exemplo: **select** Deptos **where** chefe.cidade.popul > 1 milhão.

Neste caso, são percorridas três classes, de acordo com a modelagem da Figura V.3 (pág. 62), na seguinte ordem: Dept, Func e Cidade. A classe Dept está associada à coleção Deptos e é percorrida em paralelo. As classes Func e Cidade possuem os atributos cidade e popul respectivamente. Entretanto, o acesso aos objetos das classes Func e Cidade é concorrente e eventualmente compartilhado.

A estratégia descendente nem sempre é a mais interessante devido ao compartilhamento no acesso às classes dos objetos referenciados encadeadamente. Caso as cláusulas da qualificação do predicado tivessem muitos níveis de objetos a serem percorridos, haveria grande probabilidade de se ter muita concorrência no acesso às páginas dos objetos. O acesso concorrente só não haveria nos objetos da coleção alvo, uma vez que estes foram desagrupados entre os processadores.

Por outro lado, essa concorrência só é crítica no momento do acesso ao disco, já que nas consultas os objetos não são modificados e podem estar simultaneamente nos 'buffers' dos diversos processadores.

O **algoritmo descendente** de avaliação de predicados na arquitetura do PARGOA fica da seguinte maneira, para o exemplo apresentado anteriormente:

Exemplo: **select** Deptos **where** chefe.cidade.popul > 1 milhão.

Passo 1

. No Gerente de Distribuição (GD), os objetos da classe Dept são distribuídos pelos processadores com a cláusula "chefe.cidade.popul > 1 milhão".

. Nos processadores, os algoritmos locais aos nós avaliam e enviam para o GD os IDOs que qualificam a cláusula, através do mesmo algoritmo do servidor seqüencial.

Já a estratégia de avaliação de predicados **ascendente/descendente** (A/D) é adequada ao processamento paralelo, uma vez que todos os objetos das classes envolvidas poderiam ser avaliados em paralelo. Se para diversos casos, no processamento seqüencial, a análise A/D mostrou sua superioridade em relação ao desempenho da análise descendente correspondente, no processamento paralelo esta tendencia se mantém ou se mostra ainda mais eficiente.

A estratégia ascendente pode ser implementada de modo semelhante à junção parcial que impõe uma sincronização durante a execução da operação, implementada no PARBASE. Sendo assim, essa estratégia implica que o algoritmo seja executado em mais de um passo, o que não

ocorre na estratégia descendente. O algoritmo **totalmente ascendente** no PARGOA, para o mesmo exemplo anterior, seria executado da seguinte forma:

Exemplo: **select** Deptos **where** chefe.cidade.popul > 1 milhão.

Passo 1

. No Gerente de Distribuição (GD), os objetos da classe Cidade são distribuídos pelos processadores com a cláusula "popul > 1 milhão".

. Nos processadores, os algoritmos locais aos nós avaliam e enviam para o GD os IDOs que qualificam a cláusula.

Passo 2

. O Gerente de Distribuição concentra os IDOs enviados pelos processadores e distribui a lista resultante para todos os processadores. Essa lista é enviada juntamente com a lista dos objetos da classe Func local aos processadores.

. Nos processadores, os algoritmos locais aos nós executam um algoritmo semelhante à operação de junção. Os IDOs dos objetos que pertencem ao resultado da junção são enviados ao GD.

O segundo passo é repetido até que se chegue à coleção alvo.

Conforme mencionado no capítulo V, pretende-se que a próxima versão do servidor GOA, já disponha de recursos no modelo interno de objetos que viabilizem também o processamento do algoritmo ascendente.

A seguir, são apresentados os resultados de uma avaliação do desempenho do protótipo do SBDP PARGOA na máquina paralela NCP I. Conforme comentado na Seção III.4, um sistema paralelo ideal apresenta duas propriedades chave [DeWi90b]: a aceleração linear e a expansibilidade linear.

Foi realizada uma série de testes com predicados de seleção onde foram analisados os tempos de resposta medidos para variações no número de níveis de atributos da cláusula do predicado e no número de processadores utilizados na configuração da arquitetura. Os testes foram aplicados no sentido de medir-se a aceleração do PARGOA.

As coleções utilizadas para o 'benchmark' são instâncias da mesma base de dados dos testes realizados com a versão seqüencial do GOA descrita na capítulo V. Foram realizados testes sobre coleções de 100 e 1000 objetos.

Todos os testes foram repetidos para configurações de 1, 2, 4 e 8 processadores no NCP I. Os tempos foram medidos a partir do momento que o cliente submete a operação ao PARGOA

até a finalização da operação. Foram incluídas neste tempo todas as passagens da operação pelos diversos gerentes do PARGOA.

As Figuras VI.4 e VI.5, apresentam o ganho obtido com o aumento no número de processadores para alguns dos testes realizados. Cabe lembrar que o ganho é a razão entre o tempo de execução com **1** processador e o tempo de execução com **n** processadores. São apresentados os resultados para três classes de consultas. Na consulta (a), a coleção alvo contém 100 objetos sendo acessados com um predicado simples. Nas outras duas a coleção alvo possui 1000 objetos, sendo que na consulta (b), o predicado é sobre um atributo local à própria coleção, com fator de seletividade igual a 0,1% e na outra consulta (c), o predicado é sobre um atributo que referencia outra coleção que possui 100 objetos, com fator de seletividade igual a 10%.

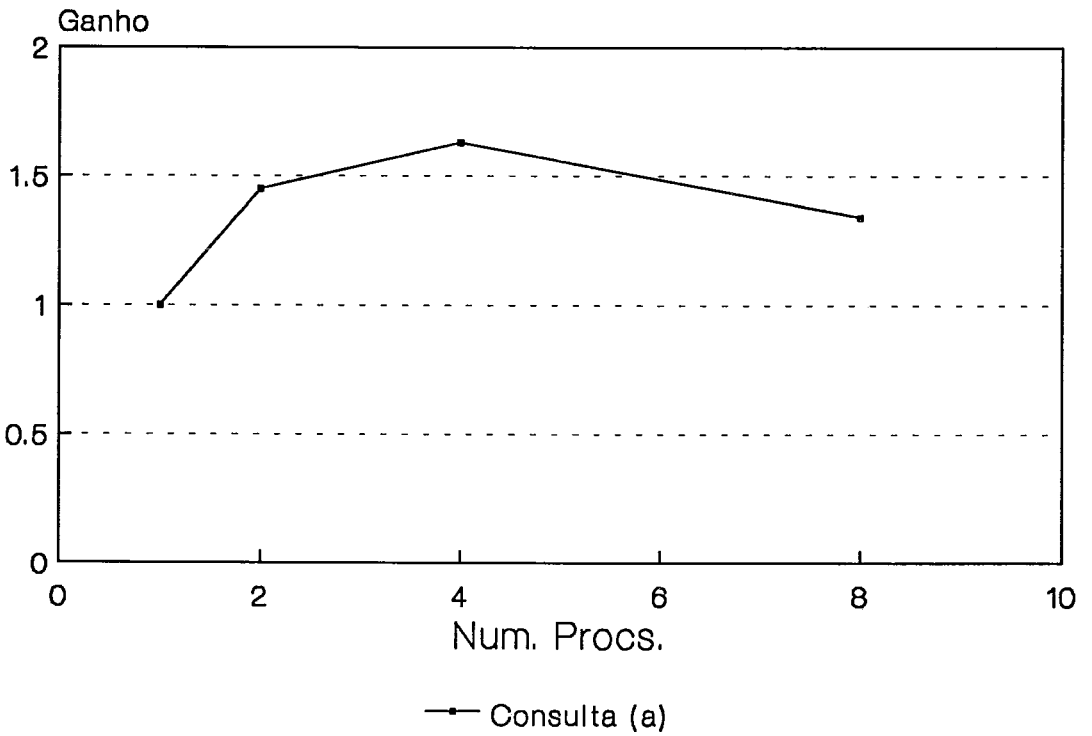

Figura VI.4 - Curva de ganho para a consulta (a) com tempo de processamento para o processador 1 = 0"645

Na consulta (a) o predicado possui um nível de atributos operando. Apesar da coleção ter sido distribuída totalmente entre os processadores, observa-se que o ganho para 8 processadores é inferior ao ganho de 4 processadores, evidenciando que o pequeno número de objetos era mais adequado à distribuição parcial. Neste caso, o Gerente de Distribuição poderia alocar somente 4 processadores para a operação deixando os demais livres para outras operações.

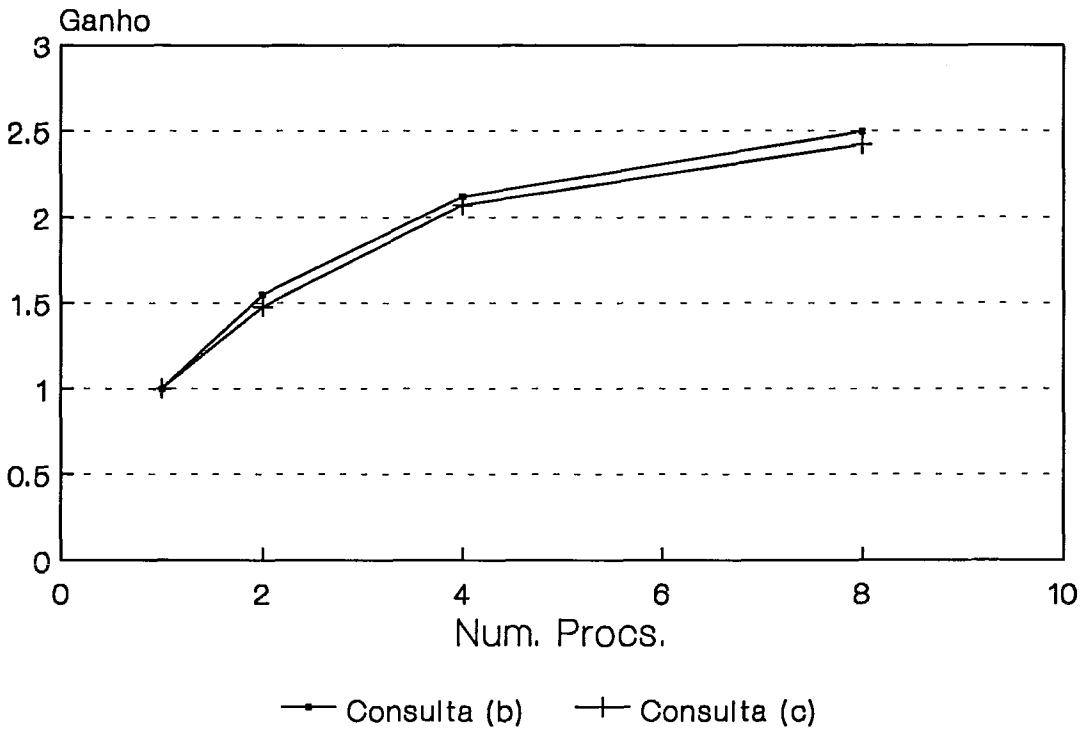

Figura VI.5 - Curvas de ganho para as consultas (b) e (c) com tempo de processamento para o processador 1 (b) = 6"336 e para o processador 1 (c) = 9"012

Observa-se que nas duas consultas sobre 1000 objetos a curva de ganho é praticamente a mesma, mostrando que embora uma das consultas tenha que acessar objetos de outra coleção concorrentemente com os outros processadores, o ganho foi mantido. O que aumentou foi o tempo de execução isolado conforme já era esperado. A manutenção da taxa de ganho evidencia que características típicas da orientação a objetos não prejudicaram o potencial de ganho do paralelismo.

É importante comentar que embora as medidas de desempenho tenham apresentado ganho em todas as situações, os resultados obtidos foram aquém do esperado. O fato do PARGOA ter utilizado um computador hospedeiro para o NCP I de capacidade inferior àquela utilizada nos experimentos do PARBASE, fez com que os testes apresentados no capítulo III fossem repetidos na configuração atual no sentido de avaliar o impacto das novas condições do hardware. De fato, obteve-se uma queda no desempenho e no ganho do processamento das consultas do PARBASE, em relação aos resultados obtidos com a primeira configuração.

Tal fato indica que embora a configuração do NCP I não tenha alterado, a mudança no computador hospedeiro, principalmente na configuração do disco rígido com características inferiores tanto na controladora quanto na capacidade de armazenamento, prejudicaram o potencial de desempenho oferecido tanto pelos algoritmos como pelos processadores do modelo T8 do NCP I.

Capítulo VII

CONCLUSÕES

A difusão dos SGBDOOs para aplicações não convencionais, vem sendo realizada com relativo sucesso, através de protótipos experimentais e de alguns produtos que começam a aparecer no mercado. A aceitação desses SGBDOOs para algumas dessas aplicações potenciais, entretanto, está condicionada ao aprimoramento de arquiteturas e de algoritmos para a gerência de objetos para que os produtos ofereçam um desempenho satisfatório.

Foram desenvolvidas neste trabalho arquiteturas que contribuem para o aumento do desempenho de SGBDOOs, através do paradigma cliente/servidor e da exploração do processamento paralelo. Para validar as arquiteturas, três protótipos foram desenvolvidos com sucesso: **i)** o servidor paralelo de consultas relacionais - **PARBASE**, **ii)** o servidor sequencial de objetos - **GOA** e, **iii)** o servidor paralelo de objetos - **PARGOA**. As implementações foram realizadas levando em conta a sua integração na construção do SGBDOO GEOTABA utilizado na Estação TABA.

Foi realizada uma caracterização de técnicas de paralelismo aplicadas a bancos de dados, que juntamente com a análise dos diversos Sistemas de Bancos de Dados Paralelos existentes, contribuiram para a avaliação do potencial que o paralelismo possui para aumentar o desempenho de sistemas de Banco de Dados.

Técnicas de paralelismo foram utilizadas em algoritmos de processamento de consultas relacionais, apresentando resultados promissores. Em diversas situações, o PARBASE obteve aceleração linear, com o tempo de processamento para uma consulta com 8 processadores sendo 1/8 do tempo total de processamento para 1 processador. A experiência realizada, mostrou que o modelo relacional é naturalmente paralelizável e que a técnica de distribuição dos dados entre os processadores utilizada é um fator determinante no ganho a ser obtido. A experiência do desenvolvimento do PARBASE, guiou a definição das diretrizes do projeto do PARGOA. Entretanto, algumas modificações tiveram que ser realizadas para a adequação ao modelo orientado a objetos do PARGOA.

Foram avaliadas diversas arquiteturas do tipo cliente/servidor para SGBDOOs e foi escolhida aquela com servidor de objetos. Esta arquitetura coloca alguma informação semântica disponível ao servidor, permitindo a interpretação dos objetos armazenados necessária à exploração do paralelismo. O servidor sequencial de objetos, GOA, desenvolvido, possui um avaliador de predicados de consulta, além das funções de gerência de armazenamento de objetos.

Foi desenvolvido no GOA, um gerente de coleções armazenadas visando a manipulação de conjuntos de objetos. O armazenamento de coleções utiliza um gerente de segmentos de páginas para controlar o agrupamento de objetos. A gerência de segmentos como unidade de armazenamento de objetos, mostrou-se bastante adequada como unidade de distribuição de objetos na implementação do PARGOA.

Diversos algoritmos de avaliação de consultas foram desenvolvidos para explorar técnicas de processamento descentente e ascendente dentro da hierarquia de classes e composição de objetos. Os testes realizados mostram que, embora pouco explorada nos trabalhos apresentados na literatura, a estratégia ascendente possui um potencial para melhorar o tempo de resposta, que deve ser considerado pelo otimizador e processador de consultas.

Foram avaliadas as dificuldades de explorar paralelismo na gerência de objetos. O servidor paralelo de objetos desenvolvido, PARGOA, apresenta soluções para conciliar a distribuição dos objetos de uma coleção, adequado ao processamento paralelo, com o requisito de agrupamento para o processamento de objetos em conjunto. A solução adotada tira proveito do modelo de memória de disco compartilhado utilizado na máquina paralela NCP I, que não obriga que os objetos estejam distribuídos fisicamente entre os processadores. Assim, o armazenamento dos objetos no PARGOA não compromete o agrupamento dos objetos nas páginas do disco.

O PARGOA utiliza um algoritmo de gerência de paralelismo para as operações implementadas no servidor de objetos e algoritmos paralelos específicos para a avaliação de predicados. A partir da aferição de tempos de resposta para diversas consultas sobre o modelo de dados orientado a objetos, foi evidenciado o ganho obtido no desempenho do servidor paralelo de objetos. Os resultados alcançados com o PARGOA, confirmam a viabilidade do uso de paralelismo em SGBDs orientados a objetos.

Com os resultados obtidos, observou-se que o desempenho de um SGBDOO pode realmente aumentar através do uso de um servidor de objetos paralelo em sua arquitetura, o que atinge a um dos objetivos do desenvolvimento deste trabalho. O uso de um servidor paralelo de objetos é ainda mais incentivado, dada a disponibilidade crescente de máquinas paralelas, tanto de alta capacidade como os computadores paralelos da Intel e Sequent, como através do uso de estações de trabalho com múltiplos processadores. É importante que novas técnicas sejam estudadas para tirar proveito do potencial desta nova realidade do equipamento, onde os usuários dos SGBDOOs podem ter muito a ganhar.

O servidor de objetos GOA, tanto no modo sequencial quanto no modo paralelo, foi projetado visando a sua utilização na construção do GEOTABA. Entretanto, seu uso não está limitado ao Projeto TABA, podendo ser utilizado como servidor de objetos para outros SGBDOOs. Esta integração é viável, pois o GOA possui uma interface bem definida, com comunicação cliente/servidor, através do código intermediário especificado. Além disso, o modelo adotado para a representação dos objetos no nível interno, não é dependente do modelo de objetos do GEOTABA. A incorporação do GOA a um SGBDOO seria facilmente atingida com a conversão dos objetos do formato do buffer de objetos do cliente para o formato de representação do GOA e com a utilização do código intermediário definido para a solicitação das operações de gerência oferecidas pelo GOA.

Este trabalho fez uso de tecnologias novas no contexto de Sistemas de Banco de Dados, a saber: **i)** processamento paralelo, **ii)** orientação a objetos e **iii)** comunicação cliente/servidor, para o desenvolvimento de arquiteturas para SGBDs com capacidade para suportar aplicações não convencionais. Além do uso dessas três tecnologias ainda não estar solidificado no contexto de banco de dados, a sua combinação tornou este trabalho uma tarefa desafiante. Os resultados obtidos com os diversos experimentos realizados, no entanto, confirmam a presença do processamento paralelo no futuro dos Sistemas de Bancos de Dados de Alto Desempenho.

Trabalhos relacionados, também vêm adotando o paradigma de cliente/servidor para arquiteturas de SGBDOOs. Entretanto, as arquiteturas que incorporam paralelismo nos SGBDOOs [Vald90, Mits92], ainda utilizam o modelo relacional na gerência e manipulação dos objetos armazenados.

Futuramente, pretende-se dar continuidade ao trabalho desenvolvido explorando os protótipos PARBASE, GOA e PARGOA. Assim como o desenvolvimento do PARBASE e do GOA forneceram subsídios para o projeto do PARGOA, planeja-se a experimentação de novos algoritmos no contexto do paralelismo do PARBASE e no contexto da orientação a objetos do GOA. Desta forma, é possível avaliar efeitos isolados das duas tecnologias, antes de sua combinação no PARGOA.

A curto prazo, tenciona-se repetir os testes realizados com o PARGOA, na nova arquitetura do NCP I através do novo disco adquirido, atualmente em fase de testes na arquitetura piloto. Com a nova realidade de espaço de armazenamento da memória secundária, pretende-se aumentar a base de testes e utilizar 'benchmarks' já definidos para a orientação a objetos [Catt92].

Como a implementação do PARGOA não está restrita ao ambiente NCP, planeja-se a utilização de redes de estações de trabalho, através do agrupamento de algumas estações que simulam um computador paralelo. Existem vários softwares que gerenciam grupos de estações de trabalho para a simulação de um hipercubo [Hami92] ou de outras ligações [Sund90]. Nesse novo ambiente, um grupo de estações seria alocado ao servidor paralelo de objetos enquanto que as outras estariam sendo utilizadas como os clientes.

O fato dos três protótipos estarem disponíveis permite que várias experiências novas sejam realizadas e sobre plataformas reais de desenvolvimento. Em relação ao processamento de consultas, pretende-se desenvolver novos algoritmos na avaliação dos predicados através da estratégia ascendente e tenciona-se explorar a integração do otimizador de consultas com o servidor de objetos, tanto no modelo sequencial quanto no paralelo. Outra contribuição nesse sentido está no estudo de viabilidade e implementação de indíces parciais de acesso associativo às coleções no contexto do PARGOA, já que os índices contribuem para um bom balanceamento de carga entre os processadores.

Pretende-se melhorar o desempenho e a funcionalidade do PARGOA através de experiências de alocação de operações aos processadores e de implementação dos algoritmos de controle de concorrência dos 'buffers' dos processadores.

Finalmente, este trabalho abre as portas de uma área de pesquisa que visa atender à necessidade de uma nova classe de sistemas de bancos de dados orientados a objetos para domínios de aplicação onde o fator desempenho é imperativo.

Referências Bibliográficas

[Aalb91] Aalbersberg,I.J. Sijstermans,F. "High-Quality and High-Performance Full-Text Document Retrieval: the Parallel InfoGuide System" Proceedings First International Conference on Parallel and Distributed Information Systems, Miami, dezembro 1991, pp.142-150.

[Agha86] Agha,G. "Actors: A Model of Concurrent Computation in Distributed Systems" MIT Press, 1986.

[Agui92] Aguiar,T.C. "XAMÃ: o Planejador de Ambientes da Estação TABA" Dissertação de Tese de Doutorado, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, março 1992.

[Agra88] Agrawal,R. Jagadish,H.V. "Multiprocessor Transitive Closure Algorithms" Proceedings of the IEEE International Symposium on Databases in Parallel and Distributed Systems", Austin, dezembro 1988, pp. 56-66.

[Alex88] Alexander,W., Copeland,G. "Process and Dataflow Control in Distributed Data-Intensive Systems" Proceedings ACM SIGMOD International Conference on Management of Data, Chicago, junho 1988, pp.90-98.

[Alma89] Almasi,G.S., Gottlieb,A. "Highly Parallel Computing" Benjamin Cummings, 1989.

[Amer89] America,P. "Issues in the design of a parallel object-oriented language" POOL2/PTC Distribution Package, Philips Research Laboratories, University of Amsterdam, março 1989.

[Amer91] America,P. "Programmer's Guide for POOL2" POOL2/PTC Distribution Package, Philips Research Laboratories, University of Amsterdam, janeiro 1991.

[Amor91] Amorim,C.L. Citro,R. Souza,A. Chaves,E. "The NCP I Parallel Computer System" Relatório Técnico ES-241/1991, COPPE-Sistemas/UFRJ, abril 1991.

[Arau91] Araujo,L.F. Moura, A.M.C. "Gerenciamento Fisico de Objetos num Sistema de Banco de Dados" Anais VI Simpósio Brasileiro de Banco de Dados, Manaus, maio 1991, pp.208-222.

[Arti90] Artificial Intelligence Limited "STRAND^{88} User Manual" Buckingham Release, junho 1990.

[Atki90] Atkinsons, M. et al. "The Object-Oriented Database System Manifesto", *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, Atlantic City, NJ, maio 1990.

[Banc88] Bancilhon F. - "The Design and Implementation of O2, an O-O Database System". *Proc of the II International Workshop on O-O DB Systems*, 1988.

[Banc89] Bancilhon, F. "Query Languages for Object-Oriented Database Systems: Analysis and a Proposal" Anais IV Simpósio Brasileiro de Banco de Dados, Campinas, SP, abril 1989, pp.22-38.

[Banc92] Bancilhon, F. Delobel,C. Kanellakis,P. (editores) "Building an Object-Oriented Database System: The Story of O_2" Morgan Kaufmann Publishers, 1992.

[Bell90] Bellosta-Tourtier,M.J. et al. "GEODE: Concepts and Facilities" 6^{emes} Journées Bases de Données Avancées, 1990.

[Berg91] Bergsten,B. Couprie,M. Valduriez,P. "Prototyping DBS3, a Shared Memory Parallel Database System" Proceedings First International Conference on Parallel and Distributed Information Systems, Miami, dezembro 1991, pp.226-234.

[Bern87] Bernstein, P. A.,"Database Support for Software Engineering --An Extending Abstract--", (th International Conference on Software Engineering, Monterey, California, março, 1987.

[Bers88] Bershad,B. Lazowska,E. Levy,H. "PRESTO: A System for Object-Oriented Parallel Programming", Software Practice and Experience, vol.18(8), agosto 1988, pp.713-732.

[Bhid88a] Bhide,A., Stonebraker, M. "A Performance Comparison of Two Architectures For Fast Transaction Processing" Proceedings of the 4th International Conference on Data Engineering, IEEE, Los Angeles, fevereiro 1988.

[Bhid88b] Bhide,A. "An Analysis of Three Transaction Processing Architectures" Proceedings of the 14th International Conference on Very Large Data Bases, Los Angeles, 1988, pp. 339-350.

[Bitt83] Bitton,D. Boral,H. DeWitt,D. Wilkinson,K. "Parallel Algorithms for the Execution of Relational Database Operations" ACM Transactions on Database Systems, v.8(3), setembro 1983, pp. 325-353.

[Bitt88] Bitton,D. Gray,J. "Disk Shadowing" Proceedings of the 14th International Conference on Very Large Data Bases, Los Angeles, 1988, pp. 331-338.

[Blum89] Blum, H. Gonçalves, L.A.C.B. Calland, L. Rossatto, M. A. Mattoso, M.L.Q. "CPRELOBJ - Um protótipo de sistema de gerência de banco de dados orientado a objetos " Anais XV CLEI, vol. 1, Santiago, Chile, julho, 1989, pp.37-56.

[Bora88] Boral,H. "Parallelism and Data Managgement" Proceedings of the 3rd International Conference on Data and Knowledge Bases, Jerusalem, Israel, junho 1988, pp.362-373.

[Bora90] Boral,H. et al. "Prototyping Bubba, A Highly Parallel Database System" IEEE Transactions on Knowledge an Data Engineering, v.2(1), março 1990, pp. 4-24.

[Borr88] Borr,A.J. Putzolu,F. "High Performance SQL Through Low-Level System Integration" Proceedings ACM SIGMOD International Conference on Management of Data, Chicago, June 1988, pp.342-349.

[Bret89] Bretl, R. et al., "The GemStone Data Management System", *Object-Oriented Concepts, Databases, and Applications*. W. Kim and F. Lochovsky Eds. ACM and Addison-Wesley, 1989.

[Brum88] Brumfield,J.A. Miller,J.L. Chou,H-T. "Performance Modeling of Distributed Object-Oriented Database Systems" Proceedings of the IEEE International Symposium on Databases in Parallel and Distributed Systems", Austin, dezembro 1988, pp. 22-32.

[Bult89] Bultzingsloewen,G. "Optimizing SQL Queries for Parallel Execution" ACM SIGMOD RECORD v.18 (4), dezembro 1989.

[Butt91] Butterworth,P. Otis,A. Stein,J. "The GemStone Object Database Management System" Communications of the ACM v.34 (10), outubro 1991, pp. 64-77.

[Care86] Carey,M. DeWitt,D. Richardson,J. Shekita,E. "Object and File Management in the EXODUS Extensible Database System" Proceedings of the 12th International Conference on Very Large Data Bases, Kyoto, agosto 1986, pp. 91-100.

[Care88] Carey,M. DeWitt,D. Vanderberg,S. "A Data Model and Query Language for EXODUS" Proceedings ACM SIGMOD International Conference on Management of Data, Chicago, junho 1988, pp.413-422.

[Care89] Carey, M. et al., "Storage Management for Objects in EXODUS", *Object-Oriented Concepts, Databases, and Applications*. W. Kim and F. Lochovsky Eds. ACM and Addison-Wesley, 1989.

[Care91] Carey,M. Franklin,M. Livny,M. Shekita,E. "Data Cashing Tradeoffs in Client-Server DBMS Architectures" Proceedings ACM SIGMOD International Conference on Management of Data, Denver, Colorado, maio 1991, pp.357-366.

[Catt91] Cattell,R.G.G. "Object Data Management" Addison-Wesley ,1991.

[Catt92] Cattell,R.G.G. Skeen,J. "Object Operations Benchmark" ACM Transactions on Database Systems, v.17(1), março 1992, pp. 1-31.

[Ceri84] Ceri,S. Pelagatti,G. "Distributed Databases: Principles and Systems" McGraw-Hill, 1984.

[Chang89] Chang,E.E. Katz,R.H. "Exploiting Inheritance and Structure Semantics for Effective Clustering and Buffering in an Object-oriented database system" Proceedings ACM SIGMOD International Conference on Management of Data, Portland, junho 1989, pp.110-121.

[Chei88] Cheiney,J-P. Maindreville,C. "A Parallel Transitive Closure Algorithm Using a Hash-Based Clustering" INRIA Research Repport 950, dezembro 1988.

[Chen76] Chen, P.P. "The Entity-Relationship Model - Toward a Unified View of Data". *ACM Trans. Database Syst.* 1:1, pp. 9-36, março 1976.

[Chen91] Cheng,J.R. Hurson,A.R. "On the Performance Issues of Object-Based Buffering" Proceedings First International Conference on Parallel and Distributed Information Systems, Miami, dezembro 1991, pp.30-37.

[Chou85] Chou,H. DeWitt,D. Katz,R. Klug,A. "Design and Implementation of the Wisconsin Storage System" Software Practice and Experience, v.15(10), outubro 1985, pp.943-962 .

[Cope88] Copeland,G. Alexander,W., Boughter,E., Keller,T. "Data Placemente in Bubba" Proceedings ACM SIGMOD International Conference on Management of Data, Chicago, junho 1988, pp.99-108.

[Dart87] Dart, S. A. et alli "Software Development Environments" IEEE Computer, novembro 1987, pp.18-28.

[Degr88] Degrazia C., Mattoso, M.L.Q. "Gerência de Versões e Controle Operacional de Objetos no GEOTABA", Relatório Técnico do Programa de Engenharia de Sistemas, COPPE/UFRJ, novembro 1988.

[Deux90] Deux, O. et al., "The Story of O2", *IEEE Trans. on Knowledge and Data Engineering*, Vol. 2, N. 1, March 1990.

[Deux91] Deux,O. et al. "The O_2 System" Communications of the ACM v.34 (10), outubro 1991, pp. 34-48.

[DeWi86] DeWitt,D. Gerber,R. Graefe,G. Heytens,M., Kumar,K. Muralikrishna "GAMMA - A High Performance Dataflow Database Machine" Proceedings of the 12th International Conference on Very Large Data Bases, Kyoto, agosto 1986, pp. 228-239.

[DeWi88] DeWitt,D.J. Ghandeharizadeh,S. Schneider,D. "A Performance Analysis of the Gamma Database Machine" Proceedings ACM SIGMOD International Conference on Management of Data, Chicago, junho 1988, pp.350-360.

[DeWi90a] DeWitt,D. et al. "The GAMMA Database Machine Project" IEEE Transactions on Knowledge and Data Engineering, v.2(1), março 1990, pp. 44-62.

[DeWi90b] DeWitt,D. Gray,J. "Parallel Database Systems: The Future of Database Processing or a Passing Fad ?" SIGMOD Record v.19 (4), dezembro 1990, pp. 104-112.

[DeWi90c] DeWitt,D. Maier,D. Futtersack,P. Velez,F. "A Study of Three Alternative Workstation-Server Architectures for Object-Oriented Database Systems" Proceedings of the 16th International Conference on Very Large Data Bases, Brisbane, Austrália, agosto, 1990, pp. 107-121.

[DeWi92] DeWitt,D. Gray,J. "Parallel Database Systems: The Future of High Performance Database Systems" Communications of the ACM v.35 (6), junho 1992, pp. 85-98.

[Ditt86] Dittrich, K.R. "Object-Oriented Database Systems - A Workshop Report" *Proc. 5th ER Conference*, Dijon, North Holland Publ. Group, 1986.

[Dunc90] Duncan,R. "A Survey of Parallel Computer Architectures" IEEE Computer, v.23(2), fevereiro 1990, pp. 5-16.

[Fang86] Fang,M. Lee,R. Chang,C. "The Idea of De-Clustering and Its Applications" Proceedings of the 12th International Conference on Very Large Data Bases, Kyoto, dezembro 1986, pp. 181-188.

[Fish89] Fishman et al., "Overview of the Iris DBMS", *Object-Oriented Concepts, Databases, and Applications*. W. Kim and F. Lochovsky Eds. ACM and Addison-Wesley, 1989.

[Flyn72] Flynn, M. "Some computers organizations and their effectiveness" IEEE Transactions on Computers, C 21, 1972, pp. 948-960.

[Fost90] Foster I., Taylor S. "Strand: New Concepts in Parallel Programming", Prentice Hall, 1990.

[Fran92] Franklin,M. Zwilling,M. Tan,C. Carey,M. DeWitt,D. "Crash Recovery in Client-Server EXODUS" Proceedings ACM SIGMOD International Conference on Management of Data, San Diego, California, junho 1992, pp.165-174.

[Frie90a] Frieder,O. "Multiprocessor Algorithms for Relational-Database Operators on Hypercube Systems" IEEE Computer, v.23(11), novembro 1990, pp. 13-29.

[Frie90b] Frieder,O. "Fault Tolerance on a Hypercube: A Database Application" Journal of Systems Software, v.13 (1), setembro 1990, pp.25-38.

[Gang92] Ganguly,S. Hasan,W. Krishnamurthy,R. "Query Optimization for Parallel Execution" Proceedings ACM SIGMOD International Conference on Management of Data, San Diego, EUA, junho 1992, pp.9-18.

[Gard90] Gardarin,G. Valduriez,P. "ESQL: An Extended SQL with Object and Deductive Capabilities, INRIA Research Repport 1185, março 1990.

[Ghan92] Ghandeharizadeh,S. DeWitt,D. "A Performance Analysis of Alternative Multi-Attribute Declustering Strategies" Proceedings ACM SIGMOD International Conference on Management of Data, San Diego, EUA, junho 1992, pp.29-38.

[Gibb83] Gibbs, S. & Tsichritzis, D., "A Data Modeling Approach for Office Information Systems". *ACM TOIS* 1:4, pp. 299-319, outubro de 1983.

[Gibb90] Gibbs,S. Prevelakis,V. "XOS: An Overview" In Object Management, Ed. Dennis Tsichritzis, Universidade de Genebra, Suíça, julho 1990.

[Gold83] Goldberg, A. & Robson, D., *Smalltalk-80 - The language and its implementation*, New-York, Addison-Wesley, 1983.

[Gonç88] Gonçalves, L.A., Mattoso, M.L.Q. "Uma Proposta de Arquitetura para o Sistema de Gerência de Objetos do TABA", Relatório Técnico do Programa de Engenharia de Sistemas, COPPE/UFRJ, novembro 1988.

[Grae90] Graefe,G. "Encapsulation of Paralelism in the Volcano Query Processing System" Proceedings ACM SIGMOD International Conference on Management of Data, Atlantic City, EUA, maio 1990, pp.102-111.

[Gray91] Gray,J.P. Poole,F. "Object-Oriented Approach for Transputer-Based Database Systems" Information and Software Technology, v.33 (1), fevereiro 1991.

[Hami92] Hamilton, J.W. Ormsby, E.M. "Simulating Hypercubes in UNIX" Dr. Dobb's Journal, dezembro 1992, pp. 72-76.

[Hard88] Harder,T. Schoning,H. Sikeler,A. "Parallelism in Processing Queries on Complex Objects" Proceedings of the IEEE International Symposium on Databases in Parallel and Distributed Systems", Austin, dezembro 1988, pp. 131-143.

[Hart88] Hart,B. Valduriez,P. Danforth,S. "Parallelizing a Database Programming Language" Proceedings of the IEEE International Symposium on Databases in Parallel and Distributed Systems", Austin, Dec 1988, pp. 72-79.

[Heil87] Heiler, J. et al, "An Object-Oriented Approach to Data Management. Why design Databases need it?", 24th ACM/IEEE Design Automation Conference Proceedings 1987, Miami Beach.

[Heli88] Helios Group at Perihelion Software Limited "The Helios Operating System" Distributed Software Ltd, Version 1, Release 1.0, agosto 1988.

[Hira91] Hirano,Y. Satoh,T, Inoue,U. Teranaka,K. "Load Balancing Algorithms for Parallel Database Processing on Shared Memory Multiprocessors" Proceedings First International Conference on Parallel and Distributed Information Systems, Miami, dezembro 1991, pp.210-217.

[Hong91] Hong,W. Stonebraker,M. "Optimization of Parallel Query Execution Plans in XPRS" Proceedings First International Conference on Parallel and Distributed Information Systems, Miami, dezembro 1991, pp.218-225.

[Hong92] Hong,W. "Exploiting Inter-Operation Parallelism in XPRS" Proceedings ACM SIGMOD International Conference on Management of Data, San Diego, EUA, junho 1992, pp.19-28.

[Huds87] Hudson, S. E. & King, R., "Object-Oriented Database Support for Software Environments", em SIGMOD- 87 Proceedings of Association for Computing Machinery Special Interest Group on Management of Data, 1987 Annual Conference, San Francisco, 1987, SIGMOD Record V. 16, N. 3, dezembro 1987, pp.27-29.

[Huls91] Hulshof,B. "Getting Started with POOL2" POOL2/PTC Distribution Package, Philips Research Laboratories, University of Amsterdam, janeiro 1991.

[Jézé92] Jézéquel,J.M. "Parallelisme Massif et Langage a Objets: Une Approche SPMD", Relatório Técnico INRIA n.1607, fevereiro 1992.

[Kaeh83] Kaehler, Ted and Krasner, Glenn "LOOM - Large Object-Oriented Memory for Smalltalk-80 Systems", *Smalltalk-80 Bits of History, Words of Advice*, Addison-Wesley, Reading, Mass., 1983.

[Kaeh87] Kaehler,T. "Virtual Memory for an Object-Oriented Language" Tutorial: Object-Oriented Computing, Ed. Gerald E. Peterson, vol. 2: Implementations, IEEE Computer Society Press, 1987, pp.201-208.

[Kell91a] Keller,A.M. Roy,S. "Adaptative Parallel Hash Join in Main Memory Databases" Proceedings First International Conference on Parallel and Distributed Information Systems, Miami, dezembro 1991, pp.58-67.

[Kell91b] Keller,T. Graefe,G. Maier,D. "Efficient Assembly of Complex Object" Proceedings ACM SIGMOD International Conference on Management of Data, Denver, Colorado, maio 1991, pp.148-157.

[Khos88] Khoshafian,S. Valduriez,P. Copeland,G. "Parallel Query Processing for Complex Objects" Proceedings of the 4th International Conference on Data Engineering, Los Angeles, fevereiro 1988, pp.202-209.

[Kim 89] Kim,W. "A Model of Queries for Object-Oriented Databases" Proceedings of the 15th International Conference on Very Large Data Bases, Amsterdam, agosto 1989, pp. 67-75.

[Kim 90a] Kim,W. "Introduction to Object-Oriented Databases" The MIT Press, 1990.

[Kim 90b] Kim,W. "Architecture of the ORION Next-Generation Database System" IEEE Transactions on Knowledge and Data Engineering, v.2(1), março 1990, pp. 44-62.

[Laks88] Lakshmi,M.S. Yu,P.S. "Effect of Skew on Join Performance in Parallel Architectures", Proceedings of the IEEE International Symposium on Databases in Parallel and Distributed Systems", Austin, dezembro 1988, pp. 107-120.

[Laks89] Lakshmi,M.S. Yu,P.S. "Analysis of parallel processing architectures for database systems", Proceedings 1989 International Conference on Parallel Processing, vol I, 1989, pp.83-90.

[Laks90] Lakshmi,M.S. Yu,P.S. "Effectiveness of parallel processing in database systems", Computer Systems Science and Engineering, v.5(2), abril 1990, pp.73-80.

[Lamb91] Lamb,C. Landis,G. Orenstein,J. Weinreb,D. "The ObjectStore Database System" Communications of the ACM v.34 (10), outubro 1991, pp. 50-63.

[Lisb92] Lisboa Filho, J. "MANO: Linguagem de Manipulação de objetos do ProtoGEO e seu Processador" Dissertação de Tese de Mestrado, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, outubro 1992.

[Li 88] Li,K. Naughton,J.F. "Multiprocessor Main Memory Transaction Processing" Proceedings of the IEEE International Symposium on Databases in Parallel and Distributed Systems", Austin, dezembro 1988, pp. 177-188.

[Maie86] Maier, D. et al. "Development of an object-oriented DBMS.", Object-Oriented Programming Systems, Languages and Applications, Portland, OR, Sept 1986. *Proceedings SIGPLAN NOTICES*, novembro 1986.

[Matt87] Mattoso, M.L.Q., "Incorporação de Ferramentas de Apoio ao Usuário do SGBD COPPEREL" Dissertação de Tese de Mestrado, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, maio 1987.

[Matt89] Mattoso, M.L.Q., Souza J. M., Gonçalves L., Degrazia C. e Rossatto M., "GEOTABA: O Sistema de Gerência de Objetos da Estação TABA", in Relatório técnico Es 179/88 do Programa de Engenharia de Sistemas - COPPE/UFRJ, 1989.

[Matt91a] Mattoso, M.L.Q. Amorim, C.L. "Uma experiência na implementação de operadores da álgebra relacional no computador paralelo NCP I" Anais VI Simpósio Brasileiro de Banco de Dados, Manaus, maio 1991.

[Matt91b] Mattoso, M.L.Q. "Bancos de Dados e Paralelismo: uma experiência prática." Submetido para apreciação em julho 1991 e aceito para publicação na Revista de Informática Teórica e Aplicada em dezembro 1991.

[Matt91c] Mattoso, M.L.Q., "Banco de Dados e Paralelismo: Um Levantamento"; Relatório Técnico do Projeto TABA, RT-08/91, abril 1991.

[Matt92] Mattoso, M.L.Q. "Taxonomias de Ambientes de Desenvolvimento de Software: A Classificação do Ambiente TABA", Relatório Técnico do Projeto TABA, RT-08/92, fevereiro 1992.

[Matt93a] Mattoso, M.L.Q. "Documentação do Servidor de Objetos - GOA", Relatório Técnico do Projeto TABA, RT-09/93, março 1993.

[Matt93b] Mattoso, M.L.Q. Fonseca,A.A.J.S. "Considerações sobre a Comunicação de Processos em Arquiteturas de Banco de Dados", Relatório Técnico do Projeto TABA, RT-10/93, março 1993.

[Mits92] Mitschang,B. "PRIMA - A Testbed for Database Processing" Anais VII Simpósio Brasileiro de Banco de Dados, Porto Alegre, maio 1992, pp. 21-38.

[Mont91] Monte, L.C.M. "Uma Representação Gráfica para o Modelo de Objetos do GEOTABA", *anais da XVII Conferência Latinoamericana de Informática*, Caracas, julho de 1991.

[Mont92] Monte, L.C.M. Lisboa Filho, J. Mattoso, M.L.Q. Souza, J.M. "Contribuição do PROTOGEO ao Projeto do SGBDOO GEOTABA" Anais VII Simpósio Brasileiro de Banco de Dados, Porto Alegre, maio 1992, pp.125-139.

[Mont93] Monte, L.C.M. "O Modelo de Objetos do GEOTABA", Dissertação de Tese de Doutorado, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, a ser defendida em maio 1993.

[Mour92a] Moura, A.M.C. Costa,M.R. "An Interactive Query Language for an Object-Oriented Database" XII International Conference of the SCCC, Santiago, Chile, outubro 1992.

[Mour92b] Moura, L.M.M. "Taxonomia de Ambientes de Desenvolvimento de Software" Dissertação de Tese de Mestrado, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, abril 1992.

[Murp89] Murphy, M.C. Rotem,D. "Effective Resource Utilization for Multiprocessor Join Execution" Proceedings of the 15th International Conference on Very Large Data Bases, Amsterdam, agosto 1989, pp. 67-75.

[Nuss91] Nussbaum,D. Agarwal,A. "Scalability of Parallel Machines" Communications of the ACM v.34 (3), março 1992, pp.57-61.

[O2Te91] O_2 Technology, "The O_2 User's Manual" Version 3.21, dezembro 1991.

[Ozsu91] Ozsu,M. Valduriez,P. "Principles of Distributed Systems", Prentice-Hall, 1991.

[Patt88] Patterson,D.A., Gibson,G., Katz,R.H. "A Case for Redundant Arrays of Inexpensive Disks (RAID)" Proceedings ACM SIGMOD International Conference on Management of Data, Chicago,junho 1988, pp.109-116.

[Pene88] Penedo, M.H. e Riddley W.E. "Software Engineering Environment Architectures" IEEE TOSE, v.14(6), junho 1988, pp. 689-696.

[Perr88] Perry,D.E. Kaiser,G.E. "Models of Software Development Environments" IEEE Proc. 10th Int. Conf. Software Engineering, Singapure, abril 1988, pp.60-68.

[Pira90] Pirahesh,H. Mohan,C. Cheng,J. Liu,T.S. Selinger,P. "Parallelism in Relational Data Base Systems: Architectural Issues and Design Approaches" IBM Research Report RJ 7724, Almaden, CA, EUA, setembro 1990.

[Poun92] Pountain,D. Bryan,J. "All Systems Go" Byte, agosto 1992, pp.112-136.

[Rich87] Richardson,P.J. Lu,H. Mikkilineni,K. "Design and Evaluation od Parallel Pipelined Join Algorithms" Proceedings ACM SIGMOD International Conference on Management of Data, San Francisco, maio 1987, pp.399-409.

[Rich91] Richard,P. "Research at Altaïr" ACM SIGMOD RECORD v.20 (1), março 1991.

[Roch90] Rocha, A.R.C. da; Souza, J.M.; Aguiar, T.C.; "TABA: A Heuristic Workstation for Software Development", *COMPEURO 90*; Tel Aviv, Israel, maio 1990.

[Rocha90] Rocha, A.R.C. e equipe TABA "UMa visão funcional da estação TABA", Relatório Técnico COPPE-Sistemas/UFRJ dezembro 1990.

[Ross88] Rossato, M.A., Mattoso, M.L.Q. "Uma Proposta para o Modelo Interno do Sistema de Gerência de Objetos do TABA", Relatório Técnico do Programa de Engenharia de Sistemas, COPPE/UFRJ, novembro 1988.

[Rous91] Roussopoulos,N. Delis,A. "Modern Client-Server DBMS Architectures" ACM SIGMOD RECORD v.20 (3), setembro 1991, pp.52-61.

[Sala91] Salamet,P.B. Chachaty,C. Dageville,B. "Compiling Control into Database Queries for Parallel Execution Management" Proceedings First International Conference on Parallel and Distributed Information Systems, Miami, dezembro 1991, pp.271-279.

[Schi92] Schiefer,B. Theobald,D. Uhl,J. "User's Guide - SOS Release 3.2" STONE Report FZI-032, Karlsruhe, Alemanha, janeiro 1992.

[Schn89] Schneider,D. DeWitt,D.J. "A Performance Analysis of Four Parallel Join Algorithms in a Shared-Nothing Multiprocessor Environment" Proceedings ACM SIGMOD International Conference on Management of Data, Portland, junho 1989, pp.110-121.

[Shaw90] Shaw, G. Zdonik, S. "A query algebra for object-oriented databases" Proceedings of the 6th International Conference on Data Engineering, fevereiro 1990, pp.154-162.

[Sinh92] Sinha,A. "Client-Server Computing" Communications of the ACM v.35 (7), julho 1992, pp. 77-98.

[Solo92] Soloviev,V. "An Overview of Three Commercial Object-Oriented DBMSs: ONTOS, ObjectStore, and O_2" ACM SIGMOD RECORD v.21 (1), março 1992.

[Souza91] Souza J.M. e equipe TABA, "Requisitos de banco de dados da estação TABA", Documento Interno - TABA, COPPE-Sistemas/UFRJ, janeiro 1991.

[Stee89] Steer,K. "Developing and Maintaining Concurrent Databases" Technical Report UKCMG 89, AI Limited, Watford, Inglaterra, 1989.

[Ston86] Stonebraker,M. "The Case for Shared Nothing" IEEE Database Engineering, v.9(1), março 1986.

[Ston88] Stonebraker,M. "The Design of XPRS" Proceedings of the 14th International Conference on Very Large Data Bases, Los Angeles, 1988, pp. 339-350.

[Ston90] Stonebraker, M., "Third Generation Database System Manifesto", *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, Atlantic City, NJ, maio 1990.

[Ston91] Stonebraker,M. "Managing Persistent Objects in a Multilevel-Store" Proceedings ACM SIGMOD International Conference on Management of Data, Denver, Colorado, maio 1991, pp.2-11.

[Stra90] Straube, D.D. Özsu, M.T. "Queries and Query Processing in Object-Oriented Database Systems" ACM Transactions on Information Systems, v.18(4), outubro 1990, pp.387-430 .

[Stro86] Stroustrup, Bjarne *The C++ Programming Language*, Addison-Wesley, 1986.

[Sun 90] Manual SUN - Volume: Network Programming Guide, março 1990.

[Tand88] Tandem Performance Group "A Benchmark of NonStop SQL on the Debit Credit Transaction" Proceedings ACM SIGMOD International Conference on Management of Data, Chicago, junho 1988, pp.337-341.

[Tera84] Teradata Corporation "DBC/1012 Database Computer Concepts and Facilities", Document C02-0001-0111, outubro 1984.

[Trot89] Trotta,C.N.F. Mattoso,M.L.Q. Souza,J.M. "Extensões do COPPEREL para aplicações nào convencionais" Anais IV Simpósio Brasileiro de Banco de Dados, Campinas, SP, abril 1989, pp.128-139.

[Tsan91] Tsangaris,M. Naughton,J. "A Stochastic Approach for Clustering in Object Bases" Proceedings ACM SIGMOD International Conference on Management of Data, Denver, Colorado, maio 1991, pp.12-21.

[Tsan92] Tsangaris,M. Naughton,J. "On the Performance of Object Clustering Techniques" Proceedings ACM SIGMOD International Conference on Management of Data, San Diego, California, junho 1992, pp.144-153.

[Ulh 91] Uhl,J. et al. "The Object Management System of Stone - SOS Release 3.2" STONE Report FZI-027, Karlsruhe, Alemanha, outubro 1991.

[Vald84] Valduriez,P. Boral,H. "Join and Semijoin Algorithms for a Multiprocessor Database Machine" ACM trans. on Database Systems, v.9(1), março 1984

[Vald90a] Valduriez,P. "Distributed and Parallel Database Systems", Apresentação de Tutorial, 5. Simpósio Brasileiro de Banco de Dados, Rio de Janeiro, abril 1990.

[Vald90b] Valduriez,P. "Query Processing in the EDS Parallel Database System", 5. Simpósio Brasileiro de Banco de Dados, Rio de Janeiro, abril 1990, pp. 2-14.

[Vele89] Velez,F. Bernard,G. Darnis,V. "The O_2 Object Manager, an Overview" Proceedings of the 15th International Conference on Very Large Data Bases, Amsterdam, agosto 1989.

[Walt87] Waltz,D. Stanfill,C. Smith,S. Thau,R. "Very Large Database Applications of the Connection Machine System" Proceedings National Computer Conference, 1987, pp.159-165.

[Wang88] Wang,X. Luk,W.S. "Parallel Join Algorithms on a Network of Workstations" Proceedings of the IEEE International Symposium on Databases in Parallel and Distributed Systems", Austin, dezembro 1988, pp. 87-96.

[Wang91] Wang,Y. Rowe,L. "Cashe Consistency and Concurrency Control in a Client-Server DBMS Architecture" Proceedings ACM SIGMOD International Conference on Management of Data, Denver, Colorado, maio 1991, pp.367-376.

[Weik91] Weikum,G. Zabbak,P. Scheuermann,P. "Dynamic File Allocation in Disk Arrays" Proceedings ACM SIGMOD International Conference on Management of Data, Denver, Colorado, maio 1991, pp.406-415.

[Wilk90] Wilkinson,K. Neimat,M.A. "Maintaining Consistency of Client-Cached Data" Proceedings of the 16th International Conference on Very Large Data Bases, Brisbane, Austrália, agosto, 1990.

[Wils91] Wilshut,A.N. Apers,P.M.G. "Dataflow Query Execution in a Parallel Main Memory Environment" Proceedings First International Conference on Parallel and Distributed Information Systems, Miami, dezembro 1991, pp.68-77.

[Wolf91] Wolf,J.L. Dias,D.M. Yu,P.S. Turek,T. "Comparative Performance of Parallel Join Algorithms" Proceedings First International Conference on Parallel and Distributed Information Systems, Miami, dezembro 1991, pp.78-88.

[Yone87] Yonezawa,A. Tokoro,M. (Editores) "Object-Oriented Concurrent Programming" MIT Press, 1897.

[Yu 91] Yu,P.S. Leff,A., Lee,Y. "On Robust Transaction Routing and Load-Sharing" ACM Transactions on Database Systems, v.16(3), setembro 1991, pp. 476-512.