BENCHREL - BENCHMARK PARA A AVALIAÇÃO DE DESEMPENHO DE SGBD's RELACIONAIS - E PROPOSTAS DE MELHORIAS PARA O COPPEREL

Maria Antonieta Rossatto

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

Aprovada por:

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

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

Prof. Geovane Cayres Magalhães, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

ABRIL 1991

ROSSATTO, MARIA ANTONIETA
BENCHREL - Benchmark para a Avaliação de Desempenho de SGBD's Relacionais - e Propostas de Melhorias para o COPPEREL (Rio de Janeiro), 1991
VIII, 338 p., 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1991)
Tese - Universidade Federal do Rio de Janeiro, COPPE
1. Banco de Dados Relacional I. COPPE/UFRJ II. Título (série).

A G R A D E C I M E N T O S

Aos meus pais, pelo carinho, amor, paciência e apoio em todos os momentos e pela harmonia familiar proporcionada.

Ao Jano Moreira de Souza, pela orientação segura, incentivo e contribuição dada a tese.

À Marta Lima de Queirós Mattoso, pelo suporte no COPPEREL e pelas ideias preciosas.

À Ana Regina Cavalcanti da Rocha, pelos conhecimentos passados ao longo do curso que permitiram desenvolver esta tese de forma mais otimizada.

À Wanderley Bertoldi de Azevedo, pelo apoio, estímulo e confiança depositados em todos os momentos.

À Marco Antonio de Britto, Chefe do Departamento de Informática, Paulo Roberto Nunes Mandarino, Chefe da Divisão de Microinformática e Administração de Dados e Helcio Blum, Chefe do Setor de Administração de Dados, pela permissão dada a realização do mestrado.

À Angela Maria Moreira, amiga e bibliotecária da Eletrobrás, pelos seus conhecimentos e contatos que permitiram a obtenção de diversos periódicos.

À bibliotecária do NCE, Embratel, Cepel, CT e IBM pela boa vontade.

Aos representantes nacionais dos SGBD's utilizados, pelo interesse demonstrado no uso de seus produtos nesta tese.

Aos demais colegas da Eletrobrás e da COPPE pela colaboração e apoio em geral.

À Eletrobrás, pelo incentivo moral e financeiro que permitiram a realização deste curso.

À Profa. Ana Regina Cavalcanti da Rocha e ao Prof. Geovane Cayres Magalhães pela participação na banca examinadora.

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

BENCHREL - BENCHMARK PARA A AVALIAÇÃO DE DESEMPENHO DE SGBD'S RELACIONAIS - E PROPOSTAS DE MELHORIAS PARA O COPPEREL

Maria Antonieta Rossatto

Abril, 1991

Orientador: Jano Moreira de Souza

Programa: Engenharia de Sistemas e Computação

Este trabalho discute alguns aspectos, existentes na literatura, de "benchmarks" e sistemas gerenciadores de banco de dados para microcomputadores. Para tanto, são apresentados aspectos estáticos e dinâmicos envolvendo a análise de desempenho de SGBD's e requisitos a serem satisfeitos pelos "benchmarks" para avaliá-los. A partir deste estudo, é apresentada uma metodologia para escolha de "benchmarks" e um quadro comparativo da funcionalidade dos SGBD's escolhidos. Finalmente, são construídos e implementados "benchmarks" para avaliar o desempenho dos SGBD's. Os resultados gerados são analisados e permitem a identificação de gargalos no SGBD COPPEREL desenvolvido pela COPPE/SISTEMAS. Propostas de melhorias são, então, apresentadas de forma a otimizar o seu desempenho.

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

BENCHREL - BENCHMARK FOR PERFORMANCE EVALUATION OF RELATIONAL DBMS'S - AND EXTENDS TO IMPROVE COPPEREL PERFORMANCE

Maria Antonieta Rossatto

April, 1991

Thesis Supervisor: Jano Moreira de Souza

Department: Engenharia de Sistemas e Computação

This work describes some aspects of benchmarks and Database Management Systems presented in the literature. Static and dynamic aspects of performance evaluation and fundamental features of benchmarks are presented. A methodology for selection of benchmarks is described and a comparative study of the DBMS's is developed to evaluate their functionality. Finally, some benchmarks are developad and implemented to evaluate the performance of the DBMS's. The results are examined and the criticals features of the relational DBMS COPPEREL, developed by COPPE/SISTEMAS, are identified. From this study, some aspects of extending the COPPEREL are proposed to improve its performance.

ÍNDICE Pág.

CAPÍTULO I - INTRODUÇÃO 01

I.1 Motivação 02
I.2 Objetivos da Tese 04
I.3 Organização da Tese 04

CAPÍTULO II - BENCHMARKS 06

II.1 Objetivos 06
II.2 Requisitos 07
II.3 Tipos de Benchmarks 09
II.4 Dificuldades 11
II.5 Análise dos Resultados 16

CAPÍTULO III - BENCHMARKS PARA BANCO DE DADOS 20

III.1 Conceitos Básicos 20
III.1.1 SGBD's Relacionais 20
III.1.2 Técnicas de Acesso/Armazenamento 24
III.1.3 Técnicas de Organização do Espaço de Armazenamento Secundário 32
III.1.4 Benchmark Sintético 34
III.2 Análise de Desempenho de SGBD's 36
III.2.1 Parâmetros Estáticos 37
III.2.2 Parâmetros Dinâmicos 44
III.3 Tipos de Benchmarks 49
III.4 Técnicas para Otimização do Desempenho de SGBD's 51
III.5 Escolha do Benchmark 59
III.6 Metodologia para Escolha e Desenvolvimento de Benchmarks 67

CAPÍTULO IV - SISTEMAS GERENCIADORES DE BANCO DE DADOS - SGBD's 78

IV.1 Escolha dos SGBD's 78
IV.2 COPPEREL 82
IV.3 DBASE III 84
IV.4 DIALOG PLUS/C 85
IV.5 ORACLE 86
IV.6 PARADOX 88
IV.7 RBASE V 89
IV.8 ZIM 90
IV.9 Considerações sobre os produtos 92

Pág.

CAPÍTULO V - BENCHREL 96

V.1 Ambiente Operacional 96
V.2 Especificação e Projeto dos Benchmarks 101
V.3 Benchmarks de Baixo Nível Propostos 113
V.4 Apresentação dos Resultados de Desempenho e Considerações sobre os SGBD's 116
V.5 Desempenho dos SGBD's em cada Benchmark de Baixo Nível 144
V.6 Execução do Benchmark de Alto Nível e Análise de Desempenho dos SGBD's 147

CAPÍTULO VI - OTIMIZAÇÃO DO CORREREL 154

VI.1 Deficiências e Propostas de Melhorias 154

CAPÍTULO VII - CONCLUSÃO 164

REFERÊNCIAS BIBLIOGRÁFICAS 169

APÊNDICE A - SISTEMAS GERENCIADORES DE BANCO DE DADOS UTILIZADOS 205

APÊNDICE B - SISTEMAS GERENCIADORES DE BANCO DE DADOS NÃO UTILIZADOS 303

APÊNDICE C - BENCHMARKS IMPLEMENTADOS EM CADA SGBD ... 309

APÊNDICE D - APLICAÇÃO AEI 331

APÊNDICE E - RESULTADOS DE DESEMPENHO DAS TRANSAÇÕES TI NOS SGBD'S 334

CAPÍTULO I

INTRODUÇÃO

A área de computação vem se desenvolvendo intensamente nos últimos anos. O mercado de informática convive com inovações e avanço tecnológico. O desenvolvimento de novos tipos de computadores e sistemas operacionais força as empresas a escolher entre os benefícios da adoção da nova tecnologia para a instalação e a continuidade do uso dos recursos existentes. Há casos em que os novos produtos são incompatíveis com aqueles em uso na empresa.

A escolha dos produtos deve considerar, principalmente, os objetivos da empresa e dos produtos. Deve-se evitar produtos com baixo desempenho e aqueles limitados a ambientes operacionais específicos, incompatíveis com os objetivos e estrutura operacional da empresa.

Toda empresa deve possuir um ambiente completo, com ferramentas que permitam o correto funcionamento da instalação e a projeção e desenvolvimento de aplicações de forma rápida, segura e eficiente. Para tanto, é necessário avaliar o desempenho dos produtos antes de sua homologação pela empresa, de forma a evitar escolhas que prejudiquem o funcionamento da instalação em uso.

A análise de desempenho de produtos considera fatores estáticos e dinâmicos que avaliam funcionalidade e tempo, respectivamente. Uma possível abordagem para a análise de

tempo baseia-se na medição de parâmetros dinâmicos pré-estabelecidos através do uso de benchmarks.

I.1 MOTIVAÇÃO

A área de banco de dados tem se desenvolvido, nas últimas décadas, de forma marcante. Novos SGBD's (Sistemas Gerenciadores de Banco de Dados) têm sido lançados no mercado, tornando grande a quantidade de SGBD's disponíveis e forçando possíveis usuários a escolher entre produtos que se diferenciam no tipo, objetivos, características, capacidade, desempenho, funcionalidade e custo. Entretanto, dificilmente, encontra-se um SGBD "ótimo" que atende todos os requisitos e resolve todos os problemas da instalação.

A manutenção de um banco de dados está consumindo a cada dia mais recursos, as aplicações estão se tornando mais complexas, ocorrendo necessidade de avaliar o desempenho de um SGBD antes de sua homologação e utilização na empresa. Mudanças no SGBD utilizado pela instalação constituem uma tarefa custosa e complexa que envolve a migração/conversão de dados e aplicações para o novo produto, treinamento de usuários, adaptação das rotinas automáticas de segurança e proteção do ambiente operacional, e construção de novos procedimentos para carregar e gerar cópias do banco de dados. Deste modo, deve-se escolher o "melhor" SGBD para a instalação de forma a realizar o mínimo de mudanças.

Nas grandes empresas, profissionais de banco de dados são responsáveis por analisar, selecionar e gerenciar SGBD's.

São tomadas decisões que podem influenciar no desempenho das tarefas executadas pelos profissionais de toda empresa. Antes de serem tomadas estas decisões, é necessário fazer um levantamento dos SGBD's existentes no mercado e seus objetivos, fixar os objetivos e requisitos da instalação e avaliar fatores que influenciam no seu funcionamento, de forma a selecionar o produto mais adequado à empresa. As decisões devem considerar o ambiente operacional da empresa, visando selecionar um SGBD compatível com a instalação, utilizar ao máximo suas potencialidades e melhorar o desempenho de toda a instalação.

Devem, também, ser analisados os resultados de desempenho gerados a partir da análise do SGBD, e a confiabilidade, segurança e funcionalidade do produto. Em alguns casos, é interessante consultar pessoas e empresas que o utilizam a algum tempo em sua instalação, para observar as vantagens e desvantagens de sua utilização. Deste modo, homologa-se o produto que "melhor" satisfaz os requisitos de desempenho e "melhor" atende as necessidades da instalação.

Entretanto, a análise de desempenho de um SGBD não deve estar associado ao objetivo de escolher o melhor, no sentido absoluto da palavra. Deve-se interpretá-la como uma forma de escolher o melhor SGBD que satisfaça os requisitos da instalação e os objetivos da empresa.

O progrRma de Engenharia de Sist2iec; da COPPE/UFRJ deiaenvolveu através de seus pesquisadores o SGBD relacional COPPEREL Sisteme de Gerêricia de Base da Dados da COPPE baseado em Álgebra Relacional), que entrou em operação em

1983. Este produto tem sido utilizado em pesquisas na universidade e serve de base para vários projetos. Entretanto, não está completo e precisa ser otimizado de forma a incorporar características e funcionalidades disponíveis em outros SGBD's. Para tanto, é preciso identificar deficiências e propor melhorias para o mesmo.

O desenvolvimento de um benchmark para avaliar o desempenho de SGBD's Relacionais permite analisar o COPPEREL e identificar as otimizações que devem ser feitas no mesmo.

I.2 OBJETIVOS DA TESE

Esta tese tem como objetivos principais fazer um estudo conceitual sobre benchmarks, criar uma metodologia para avaliar SGBD's e utilizá-la para avaliar a funcionalidade e desempenho de SGBD's existentes no mercado, avaliar benchmarks existentes na literatura, identificar um método para escolha/desenvolvimento de benchmarks, desenvolver e implementar benchmarks, comparar os resultados gerados e, finalmente, identificar possíveis deficiências existentes no SGBD COPPEREL e propor melhorias.

I.3 ORGANIZAÇÃO DA TESE

O capítulo II contém aspectos referentes a utilização de benchmarks para avaliar o desempenho da sistemas de computação. É feito um estudo conceitual e, são

apresentados objetivos e requisitos a serem satisfeitos por um benchmark, dificuldades encontradas na análise, tipos de bechmarks e resultados de desempenho.

O capítulo III trata os aspectos do capítulo II com ênfase na avaliação de desempenho de SGBD's. São, também, propostas e descritas metodologias para analisar SGBD's, escolher e desenvolver benchmarks, e avaliar a funcionalidade de produtos de banco de dados. Para tanto, são definidos parâmetros estáticos e dinâmicos envolvidos na análise, e tipos de benchmarks.

O capítulo IV descreve os SGBD's a serem analisados pelos benchmarks. é feita uma análise funcional dos mesmos, utilizando os parâmetros estáticos estabelecidos pela metodologia apresentada no capítulo III.

O capítulo V define o ambiente operacional onde serão realizados os testes, e os motivos que levaram a escolher determinados conceitos e técnicas. Os benchmarks são, então, projetados, implementados e aplicados na análise de desempenho dos SGBD's descritos no capítulo IV. A partir dos resultados dos parâmetros dinâmicos gerados pelos benchmarks de baixo e alto nível, é comparado o desempenho dos produtos e são feitas comsiderações sobre os mesmos.

O capítulo VI identifica as deficiências existentes no SGBD COPPEREL e propõe melhorias para as mesmas.

Finalmente, o capítulo VII apresenta as conclusões, fazendo uma avaliação dos conceitos e técnicas usadas nesta tese.

CAPÍTULO II

BENCHMARKS

Neste capítulo, serão apresentados aspectos referentes a utilização de benchmarks para avaliar o desempenho de sistemas de computação.

Para tanto, serão apresentados os objetivos principais de um benchmark. Serão também discutidos os requisitos que devem ser satisfeitos por um bom benchmark.

Os tipos de benchmarks serão, então, apresentados, sendo discutidos seus objetivos e dificuldades. Em seguida, serão discutidas algumas dificuldades que podem ser encontradas na escolha e utilização de benchmarks.

Por fim, serão analisados os resultados de desempenho gerados pelos benchmarks.

II.1 OBJETIVOS

Benchmarks são programas de computador que analisam a capacidade e eficiência de sistemas de programação e de ferramentas de software e hardware.

Benchmarks são muito utilizados para comparar sistemas e seus componentes, produtos de software, computadores e partes integrantes destes. Desta forma, permitem detectar falhas, fazer avaliações de desempenho, além de auxiliar na escolha de produtos que devem compor o ambiente envolvido.

Avaliações de desempenho, geralmente, usam uma aplicação como base para comparar os produtos. O benchmark é composto por medidores de tempo e pela aplicação, e sua execução gera como resposta, parâmetros de comparação pré-definidos. São alguns parâmetros de comparação: tempo de resposta, tempo de execução, tamanho da aplicação, tempo de CPU e E/S, tempo de operação, improdutividade e produtividade.

Segundo [Nich88], benchmarks podem ser vistos como um padrão para analisar desempenho de computadores, embora não seja possível estabelecer um padrão para benchmarks. Sem utilizá-los, fica difícil escolher um computador dentre vários e comparar desempenho de sistemas diferentes.

Benchmarks analisam desempenho e geram como resposta números que podem, em muitos casos, não ser significativos. Nos casos, por exemplo, em que a unidade de medida é o MIPS (Milhões de Instruções Por Segundo) e não existe um conjunto de instruções e um benchmark padrões para MIPS, não é possível comparar com precisão MIPS em CPU's com arquiteturas diferentes [Nich88].

II.2 REQUISITOS

Segundo Nicholls, em [Nich88], um benchmark deve satisfazer a quatro requisitos, que serão descritos a seguir:

a. ser significativo, isto é, testar um fator que seja relevante para o usuário.

b. ser preciso, isto é, os resultados devem conter a medida de precisão alcançada, que deve ser relatada como parte dos resultados. Os resultados devem ser claros e sem ambiguidades para facilitar o entendimento. Segundo Tom Sawyer, em [Edel90], a garantia de precisão dos resultados de desempenho gerados por um benchmark, aumenta a credibilidade do processo de utilização de benchmarks para avaliar o desempenho de produtos.

c. ser verificável, isto é, o teste poderá ser repetido tantas vezes quanto necessário. A variação no resultado deverá ser relatada.

d. ser capaz de distinguir sistemas diferentes e produzir resultados semelhantes para sistemas similares, isto é, ser capaz de distinguir diferenças significativas entre sistemas.

Além destes requisitos, existem outros que também podem ser especificados. Dentre eles podemos citar:

a. ser fácil de usar, isto é, exigir pouco esforço de seu usuário. O esforço e os conhecimentos necessários para usar o benchmark devem ser mínimos.

b. a comparação de produtos utilizando um mesmo benchmark deve ser feita dentro das mesmas condições ambientais (configuração de software e hardware), de forma a gerar resultados de desempenho significativos e comparáveis.

c. ser objetivo, isto é, saber exatamente as pretensões de cada teste que será executado.

d. ser pequeno (não mais que uma página de código), mas capaz de gerar resultados significativos [Gilb81].

e. ser portátil, isto é, permitir a sua execução em diversas máquinas, com um mínimo de adaptação [Curn76].

f. ser simples, isto é, facilmente codificado em diferentes linguagens [Curn 76].

g. ser poderoso, isto é, permitir a seus usuários entender, comparar e controlar o desempenho de sistemas [TayY85].

h. ser geral, isto é, ser independente do ambiente operacional e aplicação envolvidos na análise. O benchmark deve ser facilmente e adequadamente utilizado em qualquer ambiente com um mínimo de adaptação, e deve possuir o mínimo de restrições de uso.

i. padronizar os resultados, isto é, padronizar unidades de medida e formatos utilizados na apresentação dos resultados, facilitando o entendimento e a comparação de resultados gerados por benchmarks diferentes.

II.3 TIPOS DE BENCHMARKS

Em [Nich88], são apresentados dois tipos de benchmarks:

a. Baixo nível

   Os componentes do sistema são analisados em detalhe.

b. Alto nível

   O sistema é analisado como um todo.

Para a construção de benchmarks de baixo nível deve-se isolar o subsistema que será testado. Entretanto, segundo [FoxR88], não é possível construir um benchmark que isole totalmente o desempenho de subsistemas. Deve-se considerar a influência de outros subsistemas envolvidos. Tal influência é, geralmente, considerada em função do tempo adicional necessário para sua execução.

Os resultados gerados por benchmarks de baixo nível não possuem valor nos casos em que as pessoas responsáveis pela avaliação do sistema desconhecem a forma como subsistemas interagem para compor o sistema. Consequentemente, a precisão e eficiência do benchmark não podem ser avaliadas.

Benchmarks de alto nível devem, então, ser utilizados para confirmar os resultados gerados, anteriormente, por benchmarks de baixo nível [Greh88].

Nos casos em que os resultados forem contraditórios, deve-se selecionar outros benchmarks de alto nível, de forma a reproduzir e comprovar os resultados. Estes benchmarks selecionados devem ser confiáveis, significativos, ter sido utilizados em outras aplicações com êxito, e possuir os mesmos objetivos. Deste modo, calcula-se o desempenho total do sistema, e assegura-se que os resultados são independentes de um benchmark em particular.

Geralmente, benchmarks de baixo nível não são portáteis. Isto se deve a variação na forma como os sistemas de computadores são implementados. Não existe um padrão para implementar sistemas e, consequentemente, não existe um

padrão para desenvolver benchmarks de forma que estes funcionem em qualquer sistema sem precisar de alterações.

Benchmarks de baixo nível que rodam em ambientes com paralelismo, devem levar em consideração o paralelismo no cálculo do desempenho de sistemas. Tais benchmarks devem tentar organizar as operações dos sistemas de forma a aproveitar ao máximo as potencialidades do paralelismo e aumentar, assim, o desempenho dos sistemas.

Obviamente, não devem ser comparados resultados de tipos diferentes, isto é, não devem ser feitas comparações entre benchmarks de alto nível e baixo nível e entre resultados obtidos nestes dois tipos de testes.

II.4 DIFICULDADES

A dificuldade na escolha ou definição do benchmark a ser utilizado é grande. Primeiramente, é preciso estabelecer o tipo de produto e os parâmetros que devem ser analisados, os objetivos da análise e os benchmarks disponíveis, seus objetivos e grau de precisão.

Em seguida, deve-se escolher um benchmark que atenda os requisitos estabelecidos anteriormente, executá-lo e analisar a significância dos resultados gerados.

A seguir, serão apresentadas algumas dificuldades inerentes a escolha e utilização de benchmarks.

a. Não deve-se fazer testes fora do escopo do ambiente em uso, pois geram resultados irrelevantes, aumentando a complexidade e dificultando a análise dos mesmos.

b. Deve-se ter cuidado na escolha do benchmark a ser usado num determinado ambiente, pois há casos onde os testes disponíveis não permitem analisar todo o escopo do problema e/ou não bastam para diferenciar sistemas.

c. Nicholls, em [Nich88], define a análise de resultados gerados por um benchmark não apenas como uma ciência, mas também como uma arte que depende do julgamento humano e evolui com a experiência. Segundo ele, seria necessário desenvolver benchmarks padrões de forma a reduzir a subjetividade da tarefa.

d. Durante uma avaliação, benchmarks permanecem fixos em termos de estágios e operações envolvidos nos testes, não permitindo que o usuário faça modificações e incorpore particularidades de sua aplicação.

e. A medida que os sistemas se tornam mais complexos, aumenta o esforço necessário para geração e validação de novos benchmarks. Há casos em que o esforço não compensa os resultados que poderão ser obtidos e a solução é a utilização de aplicações reais como testes para validação e demonstração de desempenho de sistemas complexos.

f. Benchmarks são sistemas que também precisam de manutenção. Periodicamente, é necessário reanalisar o

sistema para que sejam identificados fatores de anormalidades e resultados contraditórios entre testes.

g. A apresentação dos resultados não é padronizada em todos os benchmarks. São utilizados unidades de medida e formatos que diferem entre benchmarks. Podem ocorrer distorções nos resultados em consequência do enquadramento destes no formato utilizado, tornando-os mascarados e prejudicando a análise de desempenho dos sistemas.

h. Análises de tempo de instrução geram medidas significativas, embora específicas a aplicação. Em muitos casos, ignoram a frequência de cada instrução na aplicação.

Segundo [Sirc86], tais medidas retratam o desempenho interno de sistemas sem levar em consideração o sistema operacional, o número de I/O, o tipo de processamento, dentre outros fatores.

i. Benchmarks geram resultados mais significativos quando aplicados a sistemas mono-usuários. O tempo de resposta obtido não sofre influência da concorrência entre aplicações, o mesmo não ocorre em ambientes multi-usuários.

j. Benchmarks devem ser adaptados aos tipos de aplicações que serão analisadas. Por exemplo, aplicações científicas diferem de aplicações comerciais. Logo, alguns testes da benchmark produzem resultados

significativos em algumas aplicações e resultados irrelevantes em outras.

Segundo [Vose89], um benchmark pode concluir, por exemplo, que um produto tem desempenho ruim, embora o problema não esteja no produto, mas no benchmark. Da mesma forma, um benchmark pode gerar resultados que sugiram um desempenho notável para um produto, embora seu desempenho real não seja satisfatório.

k. A carência de informações sobre os produtos em teste, pode prejudicar a escolha do benchmark e, consequentemente, prejudicar a análise do desempenho [Curn76].

l. Benchmarks são limitados, não sendo possível utilizá-los para comparar a funcionalidade de produtos. Neste caso, deve-se utilizar métodos de comparação não automatizados que permitam analisar atributos como: facilidade de utilização, interface com usuário, funções disponíveis, restrições de uso, dentre outros fatores.

m. Resultados de desempenho de um produto, gerados por benchmarks diferentes, podem variar em até 40%, dependendo dos benchmarks utilizados [Weic84].

n. O uso de benchmarks no cálculo do desempenho de produtos é visto em muitos casos, como uma forma de simplificar a análise. São analisados somente os parâmetros finais gerados pelo banchmark e ignorados os passos intermediários. Entretanto, os números finais gerados

por benchmarks são limitados e não devem ser usados como critério único para avaliar produtos [Weic84].

A causa dos problemas que ocorrem em benchmarks pode ser os programas, os métodos de teste ou o modo como os resultados são apresentados.

Benchmarks com programas mal projetados podem ser alterados pelos otimizadores e gerarem resultados incorretos e inválidos. Estes benchmarks necessitam ser reprojetados para evitar que otimizadores distorçam e invalidem seus testes de desempenho.

Existem casos em que os métodos de teste ignoram variáveis ou componentes que influenciam no desempenho e diferem entre sistemas. Deste modo, os resultados obtidos na avaliação de sistemas diferentes podem ser contraditórios, irreais e mascararem o real desempenho dos mesmos.

Segundo [Abur88], berichmarks, em si, podem ser otimizados. Para isso, torna-se necessário utilizar unidades de medida de desempenho uniformes e sem ambiguidades, fixar opções específicos para cada berichmark e garantir que estes estão sendo cuidadosamente controlados, modificar algum teste, projetar e reger outros, projetar novos testes e alterar alguns programas.

e

II.5 ANÁLISE DOS RESULTADOS

Benchmarks produzem, geralmente, resultados de desempenho na forma de números, que precisam ser interpretados. A interpretação destes resultados nem sempre é fácil [Vose89], podendo levar a conclusões incorretas, estatísticas não significativas e resultados tendenciosos.

A seguir, será feita uma análise de resultados, visando otimizar o uso de benchmarks e assegurar a interpretação correta dos resultados.

a. Quanto a Confiabilidade

Os resultados gerados por benchmarks não são totalmente confiáveis e devem ser analisados com cuidado, principalmente quando estes estão ligados a demonstração de produtos por vendedores. Estes não fazem demonstrações utilizando benchmarks cujos resultados enquadram seus produtos numa classe inferior a de seus concorrentes.

Existem benchmarks tendenciosos que permitem a seleção de parâmetros de forma a gerar os resultados desejados. Entretanto, benchmarks permanecem como a única maneira prática de estimar o desempenho de sistemas, embora, seja preciso utilizar técnicas que eliminem a manipulação dos resultados.

Estas técnicas analisam fatores que afetam o desempenho, critérios, metodologias e ferramentas de avaliação que foram utilizadas, objetivos do benchmark, origem dos

dados testados, cosifiguraçãc utilizada, a a forma como o besichmai-k foi desenvolvido a executado.

Os resultados obtidos através desta análise podem gerar dúvidas sobre a validade dos números mostrados no relatório de desempenho do produto. Neste caso, pode ser necessário reexecutar o benchmark no ambiente do usuário, de forma a reproduzir e comprovar os resultados. Deste modo, tenta-se assegurar a validade dos dados apresentados e evitar a utilização de benchmarks tendenciosos que distorçam a realidade.

b. Quanto a corretude dos produtos em teste

Geralmente, benchmarks geram resultados de desempenho independente da resposta gerada pelo produto em teste [Vose89]. Benchmarks podem concluir que o desempenho de um produto é melhor que o de outro, pois o tempo gasto na execução de uma aplicação é inferior. Entretanto, ignoram as respostas geradas durante a execução da aplicação.

Existem casos em que o produto mais rápido gera respostas incorretas. Para assegurar a validade dos resultados de benchmarks, deve-se, primeiramente, garantir o correto funcionamento dos produtos em teste.

c. Quanto a significância

Segundo [Flem86], existem casos em que um produto é analisado por vários benchmarks. Os resultados gerados são sumarizados através do cálculo da média aritmética e obtém-se o desempenho relativo do produto.

Resultados de desempenho relativo obtidos através da média aritmética, não são significativos para gerar conclusões que permitam comparar produtos. Produtos com desempenho relativo semelhantes não, necessariamente, possuem o mesmo desempenho real.

Os problemas descritos, anteriormente, ocorrem pelo fato da media aritmética ser uma medida de tendência central, fortemente influenciada pelos valores extremos da série de resultados. A uniformização dos valores da série de resultados prejudica a representação de conjuntos que revelam tendências extremas. É, particularmente, indicada no cálculo de desempenho relativo de produtos com resultados de desempenho simétricos em relação a um valor central e, concentrados num intervalo de valores pequenos.

Fleming, em [Flem86], mostra que a utilização da média geométrica no cálculo do desempenho relativo, não possui iguais problemas. O resultado obtido é significativo e pode ser utilizado em comparações de produtos. Entretanto, deve-se ter atenção aos casos em que ocorre uma grande variação entre os resultados, pois haverá influência sobre o cálculo do desempenho relativo. Deve-se, também, considerar os resultados mínimo e máximo na comparação de produtos, isto é, estabelecer o limite inferior e superior de desempenho de cada produto em relação ao benchmark utilizado.

d. Quanto a completude

A avaliação de um produto dentro de uma instalação envolve parâmetros que precisam ser analisados de forma a cobrir diferentes aspectos de desempenho. Há casos em que um benchmark não é suficiente para avaliar todos os parâmetros pré-estabelecidos.

Segundo [Webs86], os parâmetros podem ser divididos em vários grupos, de forma que o desempenho de cada grupo possa ser avaliado separadamente por um benchmark. Cada benchmark analisará um grupo de parâmetros e fará uso de uma aplicação em particular. Entretanto, não deve-se utilizar benchmarks cujos objetivos não coincidam com aqueles estipulados pelo grupo de parâmetros que será analisado.

Muitas vezes, é necessário analisar os resultados de desempenho em relação a outros parâmetros, de forma a garantir a validade e corretude dos resultados. Neste caso, cabe ao analisador do ambiente fixar tais parâmetros e aplicá-los aos resultados gerados pelos benchmarks.

CAPÍTULO III

BENCHMARKS PARA BANCOS DE DADOS

Neste capítulo, serão apresentados aspectos referentes a utilização de benchmarks para avaliar o desempenho de Sistemas Gerenciadores de Banco de Dados (SGBD's). Para tanto, serão apresentados alguns conceitos básicos que devem ser conhecidos para facilitar entendimentos futuros.

Em seguida, serão apresentados aspectos estáticos e dinâmicos envolvidos na análise de desempenho de SGBD's, requisitos que devem ser satisfeitos pelos benchmarks para avaliá-los, e tipos de benchmarks.

Finalmente, metodologias para escolha e desenvolvimento de benchmarks são apresentadas.

III.1 CONCEITOS BÁSICOS

A análise de desempenho de SGBD's envolve conceitos na área de banco de dados e sistemas operacionais. Logo, serão apresentados, a seguir, alguns conceitos básicos que devem ser conhecidos de forma a facilitar entendimentos futuros.

III.1.1 SGBD's RELACIONAIS

A maioria dos SGBD's que estão sendo desenvolvidos e comercializados no mercado são relacionais. Grande parte

das pesquisas, também, se baseiam nas idéias relacionais para lançamento de novos trabalhos [Date87]. Com o objetivo de facilitar o entendimento das características de cada produto a ser analisado nesta tese, será apresentada, a seguir, uma descrição sucinta das principais características de um SGBD relacional [Date87].

SGBD's relacionais baseiam-se nas características do modelo relacional que facilitam a descrição da estrutura lógica de um banco de dados. A única estrutura de dados usada pelo modelo relacional é a relação, cuja definição é idêntica a definição matemática exceto que as relações das bases de dados variam no tempo, pois tuplas são inseridas, deletadas e modificadas nas mesmas.

Uma relação é definida como um subconjunto do produto cartesiano dos domínios considerados. O produto cartesiano de domínios D1, D2, ..., Dn (não necessariamente distintos), é o conjunto de todas as n-tuplas (v1, v2, ..., vn) tais que v1 pertence a D1, v2 pertence a D2, ..., vn pertence a Dn. O valor de n é o grau da relação e o número de tuplas em R é a cardinalidade.

Os membros de uma relação são chamados de tuplas e podem ser representados por (v1, v2, ..., vn), sendo que cada tupla tem n componentes.

Pode-se representar as relações por meio de tabelas bidimensionais organizadas em linhas e colunas onde cada linha é uma tupla e cada coluna é um componente da tupla. Todos os valores de uma coluna são retirados do mesmo domínio.

Cada coluna da tabela representa um atributo da relação e tem um nome que deve indicar o uso que é feito dos valores do domínio correspondente, isto é, o papel que estes valores desempenham na relação. Colunas diferentes de uma mesma tabela devem ter nomes diferentes.

Cada relação possui uma chave que é uma coluna ou uma concatenação de colunas identificando unicamente uma tupla. Uma relação pode ter várias chaves candidatas, entretanto, possui uma única chave primária.

Cada relação pode ser representada pelo seu esquema que é uma lista formada pelo nome da relação e pelos nomes das suas colunas. O esquema da relação pode representar tanto um tipo de entidade quanto um tipo de relacionamento.

Se associarmos a cada coluna o seu nome, então a "posição" de cada coluna na relação pode deixar de ser considerada relevante. Cada tupla da relação passa a ser vista como um mapeamento de nomes de colunas de seu esquema para valores dos domínios correspondentes.

A definição da estrutura de uma relação usando uma LDD ("Linguagem de Definição de Dados ") consiste em especificar no mínimo o nome da relação, os nomes de todos os seus atributos e o domínio considerado de cada atributo.

As operações sobre relações são definidas fazendo uso da álgebra relacional, que utiliza uma ou mais relações como operandos e produz uma relação como resultado. Os operandos são relações constantes, ou variáveis representando relações de grau fixo e finitas.

As operações básicas da álgebra relacional são: União, Interseção, Subtração, Seleção, Projeção, Produto Cartesiano, Divisão e Junção. Informações podem ser obtidas em [Date87].

Alguns defensores do modelo relacional argumentam que o seu maior feito é representar os dados em uma estrutura tabular, e que com essa estrutura é possível modelar qualquer tipo de dado representado por um arquivo convencional, tal como arquivo de registro. No entanto, com esta visão, o modelo relacional não difere do modelo de registro. Na realidade, o maior e mais significativo feito é que todos os relacionamentos entre registros são baseados em associações simbólicas, isto é, através de comparação de valores de campos.

Outro avanço é a natureza das operações, pois no modelo relacional manipula-se, simultaneamente, um conjunto de registros ao invés de processá-los um a um. O modelo relacional oferece, também, outras contribuições importantes como a utilização do conceito de normalização, a utilização de estruturas simples e flexíveis e de linguagens não navegacionais.

No modelo relacional, há basicamente dois tipos de objetos [Hash81]: registros (tuplas) e conjuntos de registros homogêneos (relações). No entanto, com frequência, as entidades descritas no banco de dados não podem ser descritas como uma única tupla. O mapeamento de tais objetos complexos (localizados em várias tuplas) para uma ou mais relações é feito pelo programa da aplicação.

III.1.2 TÉCNICAS DE ACESSO/ARMAZENAMENTO

As técnicas de bufferização (acesso/armazenamento) utilizadas pelos SGBD's, são fatores que exercem influência sobre o desempenho destes sistemas [Sevc81]. O espaço de armazenamento deve ser criado e gerenciado de forma a reduzir o número de acessos aos dispositivos de E/S, durante a recuperação de informação do banco de dados.

Existem várias formas de organizar os dados em disco. Estruturas e técnicas de acesso/armazenamento diferentes possuem características de desempenho diferentes. Não há uma estrutura de armazenamento única, que seja ideal para todas as aplicações.

Segundo [Date87], um bom SGBD deve suportar várias estruturas de armazenamento, deve permitir que diferentes partes do banco de dados possam ser armazenadas de formas diferentes, e deve permitir que a estrutura de armazenamento de uma parte possa mudar em função de mudanças nos requisitos de desempenho.

A escolha da estrutura de armazenamento apropriada para um banco de dados grande e complexo, pode tornar-se uma tarefa difícil. Esta tarefa envolve fatores de banco de dados que serão descritos a seguir: modo de utilização, aplicações que farão uso, frequência de uso deste pelas aplicações, dentre outros fatores.

A seguir, será apresentada uma descrição sucinta de algumas técnicas de acesso/armazenamento utilizadas pelos SGBD's.

a. Concentração ("Clustering")

Consiste em tentar armazenar fisicamente em local próximo no disco, registros relacionados logicamente, pois são frequentemente utilizados juntos [Date87]. é útil quando há "localidade" nos acessos à base de dados. Define-se uma chave de "cluster" formada pelos campos do registro que serão analisados para determinar o local de armazenamento. Um campo que possui dados comuns nas várias ocorrências são armazenados somente uma vez e guardam o número de ocorrências para permitir recuperações futuras.

Esta técnica permite aumentar o desempenho dos sistemas, através da minimização do número de acessos a disco e, redução do tempo de busca a registros e do espaço de armazenamento, pelo fato dos dados comuns estarem representados somente uma vez. Deve-se evitar atualizar os campos que compõem a chave de "cluster", pois a localização dos dados é dependente destes e mudanças podem necessitar relocar fisicamente os registros.

Segundo [Date87], um SGBD deve permitir que o administrador de banco de dados especifique diferentes tipos de concentrações para diferentes arquivos, e que modifique o tipo especificado para um arquivo em caso de alterações nos requisitos de desempenho.

b. Indexação

Um índice é um tipo especial de arquivo armazenado [Date87], onde cada entrada consiste de precisamente

dois valores: um dado e um ponteiro. O dado é o valor de algum campo do arquivo indexado e o ponteiro identifica o registro deste arquivo que tem este valor de campo.

Índices são utilizados para acesso aleatório rápido a registros físicos, embora possam facilitar, também, o acesso sequencial.

Esta estrutura de armazenamento acelera a recuperação dos registros, embora retarde as atualizações. Sempre que for necessário adicionar um novo registro ao arquivo indexado, uma nova entrada deve ser adicionada ao arquivo de índices. O número de índices usados pelo sistema influencia diretamente o custo de manutenção (inserção/remoção e atualização do arquivo físico).

O administrador de banco de dados deve pesar a importância da atualização e da recuperação de registros em cada aplicação. Dependendo da aplicação, pode não ser conveniente a utilização desta técnica.

Os índices podem ser implementados através de diversas estruturas de armazenamento, sendo as principais: tabelas de espalhamento e árvores n-árias (principamente, variações de árvore-B), que serão descritas a seguir. É possível em certos SGBD's (Ingres, por exemplo) que a própria estrutura do índice seja usada para armazenar os dados indexados.

i. Tabelas de Espalhamento ("Hashing")

É um método de acesso em que os registros são armazenados no banco de dados em endereços definidos

através de uma função de espalhamento aplicada ao(s) campo(s) do arquivo pelo(s) qual(is) se quer indexar. Consiste em acessar de forma rápida e direta um registro armazenado, segundo um valor de campo de indexação [Date87].

Segundo Knuth [Knut73], este método de acesso possui desempenho superior em velocidade e espaço aos demais métodos de indexação.

Entretanto, valores diferentes para o campo indexado podem gerar endereços iguais e ocasionar o que chamamos de colisão. Deve-se selecionar uma função de espalhamento que gere endereços aleatórios bem distribuídos, reduzindo as colisões.

é teoricamente impossível definir uma função de espalhamento que gere endereços aleatórios, a partir de valores não aleatórios do campo de indexação dos arquivos do banco de dados. Consegue-se, no entanto, definir funções de espalhamento que gerem endereços com um número baixo de colisões.

Existem diversos métodos para resolver o problema das colisões. Estes consistem em escolher endereços alternativos para colocar os registros que colidiram. Dentre os métodos disponíveis, podemos citar: encadeamento separado, composto ou aberto e re-espalhamento. Informações adicionais podem ser obtidas em [Knut73].

Embora os métodos de indexação por espalhamento proporcionem excelente desempenho médio para acessos aleatórios, a maioria tem pior caso muito ruim. Para situações onde se necessite pior caso garantido, tais como em aplicações de tempo real (por exemplo, aplicações de controle de tráfego aéreo), algoritmos de espalhamento limitado foram desenvolvidos em [Souz78].

ii. Árvores n-árias

é um método de acesso em que os registros são armazenados no banco de dados em posições encadeadas de memória. O encadeamento dos registros na memórias segue a estrutura de uma árvore onde cada nó corresponde a um registro armazenado e tem n filhos [Knut73].

Segundo Knuth [Knut73], nem todas as estruturas de árvores n-árias são ideais para recuperar informações de arquivos em disco/fita, por precisarem de muitos acessos a disco para fazer uma busca. Deve-se usar, preferencialmente, variações de árvore-B que possuem os nós grupados em páginas. Deste modo, cada acesso recupera uma página de informação e cada página contém informações de vários registros, reduzindo o número de acessos a disco necessários para buscar um registro.

Inserções, deleções e atualizações de registros no arquivo tornam-se mais simples e rápidas devido a forma como a estrutura de armazenamento dispõe os

registros na memória. Entretanto, não deve-se criar páginas muito grandes pelo fato da memória principal ser limitada em tamanho e do tempo gasto na leitura da página ser alto.

Este método de acesso mantém o arquivo sempre ordenado, facilita a inserção/ remoção de registros e permite o crescimento dinâmico do arquivo. Contudo, gasta mais espaço de memória pelo fato de utilizar apontadores.

c. Técnicas de compressão

Técnicas de compressão reduzem o espaço de armazenamento necessário a um conjunto de dados e, consequentemente, reduzem o número de operações de entrada/saída necessárias para acessar um registro e aumentam a taxa efetiva de transmissão do disco para a memória principal [Date87]. Entretanto, exigem operações de CPU para comprimir e descomprimir os dados no armazenamento e na recuperação.

Dentre as técnicas de compressão disponíveis, podemos citar: supressão de caracteres repetidos, utilização de códigos de tamanho fixo mais curtos, codificação de itens frequentemente utilizados e método de Huffman. Informações adicionais podem ser obtidas em [Date87].

d. Memória Persistente

Não é suficiente, para um SGBD, que este possua uma linguagem de programação já que, nestas linguagens, os

dados manipulados são temporários. É necessário manter-se alguns dados persistentes, para que estes possam ser manipulados por seus usuários durante todo o ciclo de vida do banco de dados.

e. Identificação

Consiste em identificar os registros através do uso de um identificador que identifica univocamente um registro no banco de dados, independente de seu tipo.

A chave é um tipo de identificador e é constituída por campos que identificam univocamente um registro. No pior caso, todo registro constitui uma chave. Entretanto, ela deve ser mínima. Logo, um registro pode possuir várias chaves candidatas, mas somente uma chave primária.

As chaves e os mecanismos baseados em valores de atributos são limitados quando utilizados para identificar permanentemente um objeto [Codd79]. Isto se deve a algumas dificuldades como: os valores das chaves são determinados pelos usuários que muitas vezes precisam modificá-los em caso de alterações no banco de dados; relações diferentes que denotam a mesma entidade podem ter chaves definidas sobre domínios distintos; e pode ser necessário propagar informações de uma entidade antes de associar a esta um valor de chave.

Em [Codd79], é proposto a criação de domínios de entidade que contém os identificadores (surrogate). Usuários do banco de dados não tem controle sobre os

valores dos identificadores. A identificação de um registro é única e permanente dentro de todo o banco de dados. Assim, dois identificadores só poderão ser iguais se, e somente se, eles denotarem o mesmo registro para o sistema.

O uso de uma identificação a nível de cada registro no banco de dados, independente da classe a que pertença, permite referenciá-lo independentemente das características específicas de uma classe. A identificação do registro é a unidade que viabiliza o mapeamento do modelo conceitual para o modelo interno.

f. Gerência de memória ("Buffering")

Consiste em gerenciar o armazenamento físico dos registros na memória. Envolve políticas de gerência de memória, que determinam quando um trecho da memória secundária deve ser copiado para a memória principal.

Esta técnica depende da organização do espaço de armazenamento secundário (paginação, segmentação, ...).

Sempre que um bloco é lido do disco para a memória principal, vários processos tendem a fazer uso de instâncias comuns localizadas no mesmo bloco. Assim, as informações que necessitam ser acessadas tendem a ser as mesmas. Neste caso, geralmente, é executado o processo que necessita de um número menor de acessos a disco (aquele que possa ser executado sem necessitar de acessos adicionais a disco).

III.1.3 TÉCNICAS DE ORGANIZAÇÃO DO ESPAÇO DE ARMAZENAMENTO SECUNDÁRIO

As técnicas de organização do espaço de armazenamento secundário usadas pelo sistema operacional da máquina hospedeira, também, influenciam no desempenho do SGBD. A seguir, será apresentada uma descrição sucinta de algumas destas técnicas.

a. Paginação

É uma técnica de organização do espaço de armazenamento secundário em que a memória secundária é considerada uma extensão da memória principal, sendo abolida a distinção entre os dois níveis de armazenamento.

Na paginação, a memória secundária é dividida fisicamente em páginas de igual tamanho e a memória principal também é dividida em blocos de mesmo tamanho (igual ao da página). Esses blocos são compartilhados pelos processos, de forma que em qualquer momento, um determinado processo terá algumas páginas residentes na memória principal (páginas ativas) e as restantes na secundária (páginas inativas).

Este mecanismo possui duas atribuições que são:

i. executar a operação de mapeamento, isto é, determinar qual a página referenciada e em que bloco de memória ela se encontra.

ii. transferir páginas da memória secundária para os blocos da memória principal e guardá-las de volta na memória secundária quando não estiverem em uso.

O desempenho do SGBD fica influenciado pelo número de páginas ausentes na memória principal, e pelas políticas de remoção/substituição de páginas na memória principal utilizadas pelo sistema operacional.

b. Segmentação

É uma técnica da organização do espaço de armazenamento secundário em que a memória secundária também é considerada uma extensão da memória principal, sendo abolida a distinção entre os dois níveis de armazenamento.

Na segmentação, a memória secundária é dividida em segmentos lógicos de diferentes tamanho. Cada segmento corresponde a um procedimento, a um módulo de programa ou a uma coleção de dados. Os segmentos são carregados na memória principal, permitindo que esta reflita a estrutura dos programas.

Um segmento da memória secundária que está alocado em posições contíguas, nunca pode ultrapassar o tamanho da memória principal.

A segmentação pode ser implementada facilmente usando-se pares de registradores para demarcar os segmentos. O desempenho do SGBD fica influenciado pelo número de segmentos ausentes na memória principal e pelas

políticas de remoção/substituição de segmentos na memória principal utilizadas pelo sistema operacional.

c. Segmentação com paginação

é a técnica de segmentação implementada segundo a técnica de paginação que é aplicada em cada segmento, isto é, cada segmento é dividido fisicamente em páginas de igual tamanho.

Não é necessário manter todas as páginas de um segmento na memória principal, apenas aquelas usadas correntemente, e as páginas não precisam ser carregadas em áreas contíguas. Deste modo, um segmento não fica limitado pelo tamanho da memória principal disponível para um determinado processo.

III.1.4 BENCHMARK SINTÉTICO

São modelos utilizados para avaliar o desempenho de SGBD's baseados em suposições sobre o conteúdo do banco de dados, a localização dos dados e as aplicações em uso.

A seguir, serão descritas algumas suposições levadas a efeito por um benchmark sintético [Bitt83, Bora84, Chri84, DeWi85, Sevc81, Ston85, Turb87, Turb89].

1. Os valores dos atributos são uniformemente distribuídos pelos registros do arquivo.

2. Os atributos são independentes, isto é, o valor de um atributo não depende do valor de outro atributo.

3. As consultas realizadas pela aplicação são uniformemente distribuídas pelos atributos e registros.

4. Em arquivos blocados, cada bloco contém o mesmo número de registros.

5. As tuplas são de tamanho fixo.

Em [Bitt83, Bora84, Chri84, Dewi85,Sevc81, Ston85, Turb87, Turb89], o benchmark faz uso de um banco de dados sintético. Este tipo de banco de dados pode ser, facilmente, gerado através de programas e, simplifica a especificação de consultas e o controle sobre o tamanho das relações resultantes das consultas. Além de conseguir uniformidade nos atributos, o banco de dados permite que usuários inexperientes entendam a estrutura das relações e a distribuição dos valores de atributos.

Entretanto, as suposições, descritas anteriormente, são frequentemente, utilizadas, embora não sejam satisfeitas. Existem ambientes de banco de dados em uso que contrariam tais suposições. Os resultados de desempenho gerados podem estar incorretos e não retratarem a realidade do SGBD.

Todos os dados a serem consultados em aplicações reais não são uniformemente distribuídos. Logo, os resultados de desempenho sintéticos não devam ser comparados com resultados reais [DeWi85]. O desempenho de aplicações reais executadas em banco de dados reais difere do desempenho de testes sintéticos executados em bancos de dados sintéticos. A discrepância entre o desempenho real e o estimado pode

levar à conclusões erradas e à decisões conflitantes com os produtos e o ambiente operacional em uso.

III.2 ANÁLISE DE DESEMPENHO DE SGBD'S

Um SGBD deve garantir consistência, compartilhamento, controle centralizado, segurança e integridade aos dados [Date87], e fornecer um meio eficiente de acessar e modificar as informações armazenadas. Entretanto, deve, também, satisfazer requisitos de desempenho, visando atender as necessidades de desenvolvimento de aplicações e o ambiente operacional em uso.

A literatura sobre análise de desempenho de SGBD's é vasta. Não foi possível obter-se todos os trabalhos de forma a utilizá-los nesta tese. Entretanto, conseguiu-se uma grande variedade de referências bibliográficas que são apresentadas no final desta tese. Estes trabalhos serviram de base para o desenvolvimento de uma metodologia ("benchmark") para avaliação de desempenho de sete SGBD's selecionados no mercado de microcomputadores.

O SGBD é responsável por grande parte do processamento realizado na máquina hospedeira. Logo, a funcionalidade do SGBD pode influenciar e reduzir, significativamente, o desempenho da máquina hospedeira em ambientes multiusuário [Yao587]. As aplicações devem ser desenvolvidas de forma a respeitar as características específicas do SGBD e ajustar seu desempenho as limitações do banco de dados.

O desempenho de um SGBD depende de alguns fatores. Dentre eles, podemos citar: organização do espaço de armazenamento, tamanho da memória principal, hardware/ software básico e de comunicações usados pela máquina hospedeira, velocidade de processamento das consultas, linguagem de definição e manipulação de dados, complexidade das consultas, tamanho do banco de dados, e tempo de comunicação com o usuário (emissão de resultados).

Para analisar o desempenho de SGBD's, deve-se considerar parâmetros estáticos e dinâmicos. Parâmetros estáticos são, geralmente, usados para analisar funcionalidade. Parâmetros dinâmicos são, geralmente, utilizados para analisar tempo e implementados através do uso de benchmarks.

III.2.1 PARÂMETROS ESTÁTICOS

São parâmetros utilizados para avaliar a funcionalidade de SBGD's. Permite identificar e analisar características, ferramentas, componentes internos, capacidade de armazenamento, flexibilidades, facilidades, linguagens, interfaces, objetivos e outros fatores envolvidos com a concepção e utilização do SGBD.

Foram definidos alguns parâmetros estáticos que serão usados para avaliar a funcionalidade dos SGBD's. A seguir, será apresentada uma descrição sucinta destes parâmetros.

a. Estrutura física e organização do espaço de armazenamento

Um SGBD armazena grande quantidade de informações que devem estar organizadas de forma a facilitar a manipulação eficiente dos dados. A análise de desempenho de SGBD's deve determinar, então, o impacto da organização do espaço de armazenamento no desempenho da aplicação. Para tanto, é necessário conhecer as técnicas de acesso/armazenamento que são utilizadas.

b. Tipos de dados

Um SGBD comercial possui seu repertório de estruturas de acesso/armazenamento determinadas que não podem ser alteradas. Logo, os tipos de dados suportados pelo SGBD devem ser analisados de forma a verificar a capacidade de representação dos dados.

c. Dicionário de dados

Informações sobre aplicações devem ser armazenadas no dicionário de dados. Estas informações envolvem bases de dados, programas, telas, relatórios, perfis de segurança e outros dados que compõem a aplicação. A estrutura do banco de dados, também, deve ser definida no dicionário de dados, de forma a permitir que todas as aplicações utilizem as mesmas definições. Deste modo, garante-se que todos os dados estão armazenados em um único local e modificações nestes dados são realizadas uma única vez e propagadas para todas as aplicações que os utilizem.

d. Linguagem de definição e manipulação de dados

Todo SGBD deve possuir uma linguagem de fácil utilização, que permita aos usuários a comunicação com o

sistema, manipulando os dados de acordo com suas necessidades. Esta linguagem deve possuir alguns procedimentos básicos que realizem a inclusão, remoção, atualização e consulta de dados no banco de dados. Além dos procedimentos básicos, devem ser oferecidas ao usuário ferramentas que permitam formatar a entrada de dados, gerar relatórios, telas, gráficos e aplicações, dentre outros recursos.

Em SGBD's relacionais, a linguagem de definição e manipulação de dados deve implementar os principais operadores da álgebra relacional. Deste modo, fica garantida a construção de um grupo essencial de consultas imprescindíveis a uma aplicação.

e. Interface com o usuário

O SGBD deve possuir uma interface com usuário de forma a permitir a comunicação entre o usuário e o sistema. Geralmente, a interface é feita via menus e janelas. O usuário não precisa se preocupar com a sintaxe dos comandos. O controle é feito por teclas que ativam procedimentos pré-definidos.

Entretanto, a interface com usuário deve ser eficiente. Não é suficiente que o SGBD possua uma interface por menus, pois existem tipos de usuários e aplicações que não devem trabalhar com menus. É importante possuir, também, uma interface conversacional que permita ao usuário entrar com comandos e receber respostas diretamente. Assim, pode-se testar a funcionalidade de funções e comandos, e permite-se que usuários

experientes entrem com os comandos diretamente sem ter que escolher opções em menus [Mout88a].

Além de permitir ao usuário trabalhar com o SGBD de forma interativa, deve existir a possibilidade de executar um arquivo previamente programado e armazenado. Este arquivo permitirá ao usuário criar procedimentos não existentes no SGBD, guardá-los em arquivos, e executá-los sempre que necessário.

f. Integração com outros SGBD's

O SGBD deve transportar dados de/para outros sistemas. A migração de dados entre bancos de dados diferentes permitirá o aproveitamento de informações já existentes e reduzirá o número de tarefas e o tempo envolvido na criação e geração de um banco de dados. Neste caso, inclui-se a migração de dados entre bancos de dados de SGBD's diferentes.

g. Qualidade da documentação

O menu de auxílio, as mensagens de erro, os manuais fornecidos pelo sistema e a assistência técnica oferecida pelo fornecedor são fatores que facilitam sua utilização, pois instruem o usuário sobre o correto funcionamento de seus comandos e funções. Logo, a comunicação do SGBD com o usuário deve ser clara e precisa, e as mensagens devem retratar com precisão erros, solicitações ou avisos. Neste caso, SGBD's nacionais, geralmente, levam vantagem sobre os estrangeiros, pois as mensagens retornadas pelos

sistemas e os comandos executados pelo usuário são em português, facilitando a utilização dos mesmos.

h. Segurança e proteção dos dados

Um SGBD deve possuir mecanismos para garantir proteção e segurança aos dados, de forma a mantê-los sempre com consistência e integridade. Para tanto, devem ser estabelecidos níveis de acesso aos dados e regras de validação durante a inclusão/atualização dos mesmos, para garantir que somente pessoas autorizadas acessem o banco de dados e dados válidos sejam armazenados.

i. Deteção de falhas e reconstrução do banco de dados

SGBD's devem dispor de utilitários com mecanismos para recuperação do banco de dados em caso de falhas de transação (erros de programação na aplicação e falta de luz) e falhas de meio (problemas no sistema operacional, no SGBD, na CPU, ou nos dispositivos de E/S). Os objetivos destes mecanismos é restaurar os dados de um banco de dados para um estado usável após a ocorrência de falhas [Casa86]. Existem diversos tipos de reconstrução possíveis de serem utilizadas, entretanto, todas tem como objetivo deixar o banco de dados em um estado consistente, usável e íntegro.

j. Concorrência

Em ambientes multiusuário, é necessário coordenar a execução concorrente de transações que acessam dados partilhados e, consequentemente, interferem nas operações a serem realizadas. Segundo [Casa85], deve-se

garantir que em execuções simultâneas de um conjunto de transações, cada uma seja executada como se fosse a única do sistema. Para tanto, um SGBD deve utilizar técnicas de controle de concorrência para garantir a serialização da execução de transações concorrentes, de forma a garantir a execução completa de uma transação antes do início da próxima.

Uma transação deve manter o banco de dados num estado logicamente consistente e, somente, alterá-lo em execuções concluídas com sucesso. Um SGBD não deve permitir que uma transação executada com fim anormal altere os dados.

k. Portabilidade e flexibilidade

O SGBD deve ser portátil e flexível, podendo ser executado em diferentes tipos de computadores e ambientes operacionais, e permitindo que ocorram mudanças no hardware da instalação sem comprometimento dos dados e aplicações desenvolvidas sob o produto e que precisarão ser migrados para o novo ambiente.

l. Gerador de aplicações

Um gerador de aplicações deve permitir ao usuário desenvolver e alterar aplicações sem programação, com tempo e custo de desenvolvimento reduzidos. Este deve ser composto de geradores de base de dados, relatórios, telas, consultas, gráficos e outras tarefas que devam ser automatizadas. Os componentes da aplicação devem ser gerados, automaticamente, por seus respectivos

geradores, que produzem o código fonte a partir da seleção das tarefas a serem executadas em menus.

m. Ambiente operacional

O uso de um SGBD necessita de uma configuração mínima de hardware, que depende do tipo de ambiente em uso. Deve-se, também, considerar que ambientes monousuário, geralmente, necessitam de configuração diferente dos multiusuário. Para tanto, deve-se analisar requisitos operacionais do produto, como sistema operacional, quantidade mínima de memória RAM e em disco, tipos de equipamento compatíveis, controladora e periféricos indispensáveis a seu funcionamento.

Após fixar os objetivos e requisitos da instalação em uso e avaliar seu ambiente operacional, deve-se selecionar o "melhor" SGBD para a empresa, que seja compatível com os recursos disponíveis, sem, entretanto, perder sua funcionalidade e potencialidade.

Em alguns casos, torna-se necessário analisar a arquitetura utilizada na implementação do SGBD, e os utilitários disponíveis. Assim, permite-se que instalações usem a potencialidade máxima de cada produto, adaptem-o as suas necessidades e otimizem suas tarefas.

Os parâmetros estáticos ajudam a compreender e utilizar um sistema gerenciador de banco de dados. Deve-se analisar a funcionalidade de um SGBD antes de utilizá-lo.

III.2.2 PARÂMETROS DINÂMICOS

São parâmetros utilizados para avaliar o desempenho de SGBD's, através da medição do tempo e da análise dos recursos envolvidos na execução de aplicações. São implementados através do uso de benchmarks.

Benchmarks tem sido utilizados, com sucesso, na avaliação de desempenho de SGBD's [Ahn186, Ande89, Bitt83, Bitt88, Bora84, Bora86, Cole87, DeWi81, DeWi85, DeWi88, Duh188, Ede190, Hawt79, Hawt82, Hawt85, Info88a, Kras86a, Kras86b, Maga81, Mart86, Mato89, Mudi88, Pere89, Pug186a, Rube87, Rag189, Sevc81, Seym88a, Seym88b, Seym88c, Shaw88, Ston85, Stre90, Tand88, Turb87, Turb89, YaoS87].

Benchmarks representam uma tentativa de desenvolver uma metodologia para avaliar o desempenho de SGBD's. Estas metodologias diferem entre si na precisão e confiabilidade do tratamento dos parâmetros analisados [Fedo87].

Além de fornecer um mecanismo para comparar o desempenho de sistemas, benchmarks são ferramentas que podem ser utilizadas pelos desenvolvedores de SGBD's para avaliar o produto em relação àqueles existentes no mercado, detectar falhas, incorporar novas características, aumentar a funcionalidade e otimizar o desempenho. O novo produto deve ser capaz de competir no mercado, possuindo desempenho igual ou superior à dos produtos comercializados, e dispondo, preferencialmente, de características inovadoras. Entretanto, o benchmark só pode ser aplicado ao novo SGBD após a implementação e funcionamento de todos os seus

componentes, pois estes precisam ser executados, analisados e comparados com componentes de outros SGBD's.

Benchmarks que analisam e comparam SGBD's são, geralmente, testados em aplicações do mundo real. As aplicações são executadas nos diversos SGBD's, de forma a permitir que sejam comparados, eficientemente, os parâmetros pré-estabelecidos pelo benchmark. São parâmetros de comparação: tempo de resposta, tempo de execução, tempo de CPU, tempo de E/S, tempo de operação, tempo de improdutividade, produtividade, dentre outros.

Os resultados de desempenho obtidos pelo benchmark, representam o desempenho dos produtos em relação a um determinado tipo de aplicação. Estes resultados não podem ser generalizados para todos os tipos de aplicações, pois um SGBD não se comporta igual em todas elas. Logo, as aplicações testadas devem representar situações reais que ocorrem no ambiente em questão, de forma a simular o funcionamento do SGBD na instalação em uso e gerar resultados de desempenho representativos e bem próximos da realidade da empresa.

Quando existem na instalação vários tipos de aplicações que precisam ser testadas nos diversos SGBD's, a implementação e execução destas em cada SGBD pode ser uma tarefa com custo elevado. Uma alternativa seria determinar os requisitos e objetivos das aplicações que devem ser testados, construir e implementar uma aplicação simples e genérica que incorpore estas características, executá-la

nos diversos produtos, e comparar os resultados de desempenho obtidos pelo benchmark [Ande89].

SGBD's diferentes não possuem, necessariamente, a mesma linguagem de definição e manipulação de dados. Para comparar o desempenho destes, é necessário executar a mesma aplicação codificada nas diversas linguagens.

Cada linguagem possui seus recursos e características. A codificação da aplicação deve ser feita de forma otimizada, fazendo uso destes recursos, para retratar potencialidades da linguagem e gerar resultados de desempenho próximo da realidade. Entretanto, não devem ser feitas inovações durante a codificação da aplicação para torná-la mais rápida, pois o ganho obtido no tempo de execução é irreal, e não deve ser considerado na análise do desempenho.

A maioria dos SGBD's disponíveis no mercado utilizam uma linguagem de quarta geração para definir e manipular seus dados. Pesquisas [Mato89] mostram que apesar destas linguagens simplificarem o processo de comunicação homem-máquina, reduzirem o tempo de desenvolvimento e manutenção de aplicações, possuírem interface amigável, geradores de tela e relatórios, e facilidades de manipulação de dados, geram código executável menos eficiente e o tempo gasto com operações de E/S é alto. Logo, prejudicam o desempenho das aplicações, aumentando o tempo de execução destas.

Em [Mato89], foram desenvolvidas aplicações utilizando as linguagens de quarta geração presentes nos SGBD's e uma linguagem de terceira geração: pertence na máquina interface com permissão de acesso ao banco de dados.

Tempos de execução relativos a diferentes grupos de operações foram coletados nos SGBD's, utilizando os dois tipos de linguagem. Os resultados mostram que, na maioria dos casos testados, os tempos de execução das aplicações usando linguagens de quarta geração são superiores. Cabe ao administrador de banco de dados decidir quais os melhores recursos a serem usados em cada aplicação e a importância do tempo de resposta e da facilidade de uso do produto no desenvolvimento de aplicações no ambiente em questão.

Muitas vezes, o SGBD dispõe de comandos que permitem integrar programas e subrotinas desenvolvidas em linguagem de terceira geração com aplicações desenvolvidas utilizando sua própria linguagem natural. Deste modo, pode-se utilizar ambas as linguagens para codificar uma aplicação, distribuindo o código pelas linguagens de acordo com os objetivos de desempenho de cada módulo.

Existem técnicas especiais para otimizar o desempenho de SGBD's, as quais serão descritas, posteriormente, no item III.3. Entretanto, caso se deseje analisá-las durante a execução do benchmark, deve-se fazer uso destas em todos os SGBD's a serem comparados, de maneira uniforme.

A aplicação deve ser pequena e simples, de facilmente codificada nas diferentes linguagens. Nos SGBD's que não possuem alguns tipos de dados ou recursos, devem ser utilizadas rotinas de conversão de forma a substituir, precisamente, estas deficiências no sistema. Assim, garante-se a codificação da aplicação nos diversos produtos e a portabilidade do banco de dados.

Deve-se ter atenção à construção de aplicações que necessitem de funções não disponíveis em algumas linguagens, pois os resultados de desempenho podem ser prejudicados. Nos casos em que não é possível codificar subpartes de uma aplicação na linguagem do SGBD, deve-se excluí-las dos benchmarks de alto nível que testarão cada produto. Deste modo, evita-se que os resultados gerados fiquem distorcidos e insignificantes quando comparados com os de outros produtos. Deve-se incluir, somente, as transações implementáveis em todos os SGBD's num benchmark de alto nível.

A execução dos benchmarks nos SGBD's gera resultados de desempenho. Tais resultados variam com a natureza da aplicação, recursos e ambiente operacional utilizados. Logo, estes devem representar o mais fielmente possível o desempenho do SGBD, pois, geralmente, servem de base para profissionais de banco de dados escolherem seus produtos.

Entretanto, existem SGBD's que possuem resultados de desempenho muito semelhantes e funcionalidades muito diferentes. Isto se deve ao fato da maioria dos produtos disponíveis no mercado diferirem, entre si, em desempenho, facilidade de utilização e funcionalidade. Logo, é necessário analisar não somente os parâmetros dinâmicos, mas também os parâmetros estáticos para escolher um SGBD.

A análise dos parâmetros estáticos deve preceder a análise dos parâmetros dinâmicos. Uma vez conhecida a funcionalidade de um SGBD, torna-se mais fácil avaliar seu desempenho pois são conhecidos melhor disponíveis,

composiestei integrastes e foimas de utilizaçãu. Desite modo, pode-se melhor implementar uma aplicação e deienvolver um bcnchmark, otimizasido a utilizacão do SGBD e me1hor avaliasido os reiultados de dasempenho.

III.3 TIPOS DE BENCHMARKS

Os tipor da banchmarks para SGBD's sãc, baiicasiesite, os meiior tiiioi; deicritor so capítulo anterioi. São eles: baixo siivel e alto sivel.

Benchmarks de alto nível são benchmarks que analisam SGBD's através da execução de uma aplicação completa e coleta dos resultados.

Benchmarks de baixo nível são benchmarks que analisam SGBD's através da execução de partes isoladas de uma aplicação e coleta dos resultados.

Para SGBD's relacionais, a aplicação é, geralmente, dividida de acordo com as operações relacionais [Bitt83,DeWi85]. Cada parte da aplicação executada pelo benchmark de baixo nível, representa uma transação formada por uma ou mais operações relacionais.

Para cada transação, executa-se o benchmark de baixo nível, obtém-se os resultados de desempenho e compara-se os diferentes SGBD's em relação a operação analisada. O desempenho de cada operação é analisado individualmente, simplificando o entendimento dos resultados [DeWi85].

Estes resultados podem ser utilizados para validar o desempenho total do SGBD em relação a uma aplicação. Para tanto, estes são somados de acordo com a frequência de uso da transação na aplicação considerada. Os resultados obtidos tendem a representar, de forma aproximada, o desempenho total do SGBD.

Resultados de desempenho gerados por benchmarks de baixo nível, devem ser analisados cuidadosamente. O número de consultas analisadas pelo benchmark tende a ser pequeno. Logo, fica difícil determinar se os resultados obtidos são representativos do desempenho do SGBD [DeWit81].

Em alguns casos, benchmarks que analisam banco de dados são testados em aplicações do mundo real que envolvem um ambiente restrito, gerando resultados limitados. No caso de aplicações cujo ambiente é muito amplo e envolve estruturas diferentes, os resultados gerados nem sempre retratam a realidade.

O benchmark utilizado para comparar SGBD's deve ser invariante com relação as operações testadas, independente de seu tipo, não devendo ser adaptado a cada produto. Consequentemente, deve ser portátil e geral, podendo ser utilizado para testar o desempenho de vários SGBD's em vários ambientes operacionais.

Adicionalmente, o benchmark deve ser simples e fácil de utilizar. Este deve requerer um esforço mínimo do usuário para sua execução. Simplicidade não implica em perda de poder como muitas pessoas pensam. É errado pensar que para analisar um SGBD complexo, é necessário utilizar um

benchmark complexo. Existem benchmarks simples que analisam sistemas complexos de forma segura e eficiente.

III.4 TÉCNICAS PARA OTIMIZAÇÃO DO DESEMPENHO DE SGBD's

Estudos realizados [Bitt83, DeWi88, Hamm79, Hawt79, Hawt82, Hawt85, Ozka77, RobJ88, Sher76, YaoS87] mostram que o desempenho de SGBD's pode ser otimizado, com o uso de técnicas especiais. Algumas destas técnicas não se aplicam a versões para microcomputadores nem a versões monousuárias. Entretanto, é interessante apresentá-las como forma de tentar otimizar o desempenho de SGBD's, também, em ambientes multiusuário.

A otimização do funcionamento do SGBD numa instalação altera os resultados de desempenho gerados pelo benchmark. Deste modo, um benchmark executado em ambientes diferentes pode gerar resultados de desempenho distintos para o mesmo produto.

Dentre as técnicas utilizadas para melhorar o desempenho de SGBD's, podemos citar: particionamento de atributos [Hamm79], utilização de uma máquina de banco de dados [Bitt83, DeWi88, Hawt82, Hawt85, Ston88, YaoS87], utilização de um processador associativo [Ozka77], concentração das relações em disco [Hawt79], utilização de buffers [Sher76] e utilização de otimizadores de consulta [RobJ88]. A seguir, será apresentada uma descrição sucinta destas técnicas.

a, Utilização de uma **máquina de banco** de dados

Existem diversas arquiteturas para máquina de banco de dados. Entretanto, não existe um tipo padrão de máquina de banco de dados que seja ideal para executar todos os tipos de aplicações [Hawt82].

Uma máquina de banco de dados é uma máquina com hardware especial, inteiramente dedicada a operações de banco de dados. Yao, em [YaoS87], propõe um tipo desta arquitetura, como forma de separar o processamento realizado pelo SGBD da máquina hospedeira.

Os programas de aplicação geram operações de banco de dados que são interpretadas pela máquina hospedeira e transmitidas para a máquina de banco de dados, onde são executadas. Os resultados são devolvidos à máquina hospedeira.

A máquina de banco de dados fica responsável pela gerência e execução de todo o processamento realizado pelo SGBD. Logo, o desempenho da máquina hospedeira não é influenciado pela funcionalidade deste. Entretanto, o tempo de resposta do SGBD pode ser afetado pelas operações de transmissão e recepção realizadas entre a máquina hospedeira e a máquina de banco de dados. A utilização de softwares/hardwares eficientes pode minimizar o tempo adicional.

O número de máquinas de banco de dados produzidas é, infinitamente, inferior ao de SGBD's tradicionais produzidos e lançados no mercado. Deve-se considerar que a maioria dos SGBD's comercializados não podem ser

executados em hardwares especiais. Consequentemente, não funcionam em máquinas de banco de dados.

Segundo [Bitt83], máquinas de banco de dados não substituirão as máquinas convencionais. O ganho de desempenho obtido não é significativo em relação as mudanças que precisam ser realizadas na instalação.

Em algumas classes de aplicações, o grau de otimização de desempenho obtido através do uso de uma máquina de banco de dados, são justificativa a utilização de um hardware especial.

Existem diversos tipos de máquina de banco de dados. Um dos tipos, proposto em [Ozka77], é o processador associativo, que será descrito a seguir.

i. Utilização de um processador associativo

Um processador associativo pode ser considerado um tipo especial de máquina de banco de dados.

Ozkarahan, em [Ozka77], propõe o uso de um processador associativo como suporte de hardware na utilização e manipulação de SGBD's relacionais.

Os dados são armazenados diretamente como relações. Estas são acrescidas de bits, usados temporariamente para representar subconjuntos de tuplas na relação. Consequentemente, os resultados gerados por uma instrução, podem ser usados por outras instruções. Entretanto, há limitações no tamanho do banco de

dados e no tamanho e número de tuplas e atributos representados nas relações.

A execução de operações de busca de dados, utilizando este processador, é feita diretamente. Não há necessidade de softwares adicionais como nas máquinas convencionais. O processador dispõe de instruções próprias que auxiliam na execução das operações de busca de dados.

O processador utiliza as características de memória associativa, eliminando a indexação e os ponteiros necessários para estabelecer caminhos de acesso aos dados. Entretanto, a utilização de um processador associativo, sofre as maiores restrições de uso de uma máquina de banco de dados, no que diz respeito a integração deste no mercado de SGBD's.

b. Concentração das relações em disco ("Clustering")

Hawthorn, em [Hawt79], propõe a utilização da técnica de concentração para armazenar os dados no banco de dados. Contudo, esta técnica vem sendo empregada há longo tempo em sistemas não relacionais (por exemplo, o SGBD Codasyl utiliza o conceito de área).

Esta técnica consiste em tentar armazenar fisicamente no mesmo local em disco, registros relacionados logicamente, pois são frequentemente utilizados juntos [Date87].

O armazenamento de relações, que se relacionam logicamente, em páginas sequenciais de um arquivo físico em disco, aumenta o desempenho do SGBD, através da minimização do número de acessos a disco e, consequentemente, do tempo de E/S. Adicionalmente, a concentração de relações em disco reduz o tempo de posicionamento das cabeças do disco ("seek" + latência rotacional) para leituras consecutivas e aumenta a taxa de transmissão entre memória e disco.

O tempo de E/S de cada aplicação depende da localização dos dados em disco e do padrão de utilização dos dados. A aplicação deve ser desenvolvida integrando as relações e as consultas, de forma que as relações armazenadas sequencialmente em disco sejam acessadas sequencialmente pelas consultas.

c. Particionamento de atributos e relacões

Hammer, em [Hamm79], propõe a utilização da técnica de particionamento de atributos como forma de agrupar os atributos relacionados logicamente.

Esta técnica consiste em dividir os atributos de um arquivo em subarquivos armazenados separadamente. São grupados em um mesmo subarquivo, os atributos frequentemente, acessados juntos. Segundo [Date87], existem dois tipos de particionamento: vertical (baseado nos atributos) e horizontal (baseado nos valores de atributos).

Tal técnica deve ser utilizada na fase de projeto do banco de dados da aplicação. A aplicação, então, fará uso de um banco de dados mais otimizado. A definição de visões pode ter o mesmo efeito quanto ao aproveitamento de memória, mas não quanto à concentração e taxa de transferência. Outra opção é a normalização do arquivo.

Com a utilização desta técnica, consegue-se reduzir a quantidade de informação ociosa na memória principal e, consequentemente, os acessos a memória secundária.

Esta técnica deve, somente, ser utilizada quando a aplicação permitir dividir seus atributos. A aplicação não pode fazer uso constante de atributos pertencentes a grupos diferentes, pois aumentará o número de acessos a memória secundária.

d. Utilização de buffers

Sherman, em [Sher76, Bric77], e Sacco, em [Sacc86], propõem a utilização de áreas de memória principal ("buffers") para guardar registros acessados constantemente em aplicações que necessitam realizar muitas operaçoes de E/S.

O buffer consome fatias de memória principal não utilizadas. O desempenho do SGBD será otimizado pelo rápido acesso as informações na memória principal e redução no número de acesso aos dispositivos de E/S.

Entretanto, em sistemas que utilizam o mecanismo de memória virtual e as fatias de memória não utilizadas também são virtuais, um aumento no tamanho do buffer

ocasionará uma queda no desempenho do SGBD, pelo fato de aumentar a competição entre o buffer e a aplicação pela memória principal. Ocorre, então, a necessidade de organizar e gerenciar a memória de forma a evitar quedas no desempenho do SGBD.

Devem, também, ser analisados os seguintes fatores que neste contexto podem afetar o desempenho do SGBD: tamanho da memória principal, tamanho do buffer, política de substituição de páginas na memória, política de gerência do buffer e padrão de requisição de dados (sequencial ou aleatório) pela aplicação.

A utilização de buffers deve ser gerenciada de forma a otimizar o desempenho do SGBD em uso. O sucesso deste mecanismo depende da aplicação em execução no produto. Existem aplicações que quando são executadas utilizando este mecanismo, não ocorre ganho no desempenho. Neste caso, não se justifica o uso de buffers de memória.

e. Utilização de Otimizadores de Consulta

Robie, em [Rob.J88], propõe a utilização de otimizadores de consulta como forma de responder mais rapidamente as consultas de usuários.

Um otimizador de consulta escolhe a forma mais eficiente de acessar os dados a partir de informações sobre a estrutura do banco de dados. Para tanto, o otimizador precisa conhecer a distribuição dos dados nas tabelas. Um dos métodos utilizados é a distribuição por intervalos, que consiste em dividir as tabelas em

intervalos com a mesma quantidade de valores, e identificar o maior valor do intervalo.

Consultas podem ser feitas, interativamente, através de menus ou por programas. No primeiro caso, a consulta é otimizada somente uma vez, enquanto no segundo caso podem ser realizadas otimizações em tempo de compilação e execução. Alterações na distribuição dos dados em tempo de execução devem gerar novas otimizações, embora o tempo de improdutividade gasto em cada otimização prejudique o desempenho do SGBD. Aplicações com distribuição de dados relativamente estável no banco de dados, precisam ser otimizadas somente na primeira vez que forem executadas.

Para obter a forma mais eficiente de otimizar a corisulta, o otimizadoi- arialisa o custo de CPU e E/S arivolvidos em cada alternativa possível de ser adotada. O curto vai-ia com os objetivos de cada sistema. Deste modo, é escolhida a alterriativa de menor custo.

Os otimizadores de consulta teritam, também, combiriar oa; dados durante uma consulta, de foi-ma a facilitai- a recuperação dos mesmos. Para tanto, levam em consideração a ordem com que os dados são recuperados e o tamanho das tabelas, índices disponíveis, técnicas de acesso/armazenamento utilizadas pelo SGBD, e informações estatísticas sobre o conteúdo das tabelas.

O desempenho de SGBD's relacionais pode ser melhorado com a utilização de otimizadores de corisulta. Entretanto, ecte deve sei- preciso e escolher sempre a

melhor alternativa de otimização, de forma a evitar quedas de desempenho.

Estas técnicas podem ser utilizadas para otimizar o desempenho de SGBD's. Entretanto, a instalação em uso e as aplicações a serem desenvolvidas pela empresa possuem suas características e objetivos já definidos. A escolha do SGBD e das estratégias a serem utilizadas para melhorar o seu desempenho deve respeitar tais características e objetivos.

III.5 ESCOLHA DO BENCHMARK

A escolha e construção do benchmark a ser utilizado na análise de desempenho de SGBD's é uma tarefa difícil e complexa, que involve diversos fatores.

Existem várias alternativas que podem ser seguidas para utilizar um benchmark. Dentre elas, podemos citar [Ston85]:

a. Utilizar um benchmark pronto

Consiste em utilizar um benchmark já construído e testado, para analisar um SGBD. Entretanto, deve-se analisar a procedência dos benchmarks, seus objetivos e testes já realizados em SGBD's, e ter cuidado com as limitações destes que não foram construídos especificamente para atender os objetivos do ambiente em uso. Cabe ao usuário responsável pela análise, escolher, dentre os benchmarks existentes, aqueles que satisfazem as necessidades e objetivos da instalação. É

aconselhável, que sejam utilizados vários benchmarks para avaliar o desempenho de um produto, de forma a confirmar resultados.

Em [Bitt83, Bitt88, Bora84, Hawt85, Ston85] é apresentado o benchmark de "Wisconsin". Apesar deste benchmark estar se tornando um padrão para avaliar o desempenho de SGBD's relacionais, em alguns casos, ele não gera resultados de desempenho representativos da aplicação em uso.

Isto se deve ao fato deste benchmark ser sintético, isto é, utilizar dados gerados através de programas. Estes dados são gerados levando-se em consideração suposições que nem sempre são verdadeiras em ambientes reais. Além disso, este benchmark possui limitações nos tipos e estruturas de dados, operações e tamanho das relações suportadas, não possuindo, por exemplo, operações de ponto flutuante e não suportando evoluções de esquema. Estas limitações influenciam na análise de desempenho dos SGBD's.

SGBD's diferem, entre si, no suporte as operações de ponto flutuante. Alguns SGBD's simulam estas operações utilizando números decimais como mecanismo de armazenamento. Outros usam o hardware de ponto flutuante disponível na instalação. O benchmark de "Wisconsin" não é capaz de perceber esta diferença.

Muitas aplicações precisam modificar seus esquemas de banco de dados diversas vezes. Ocorre a necessidade de modificar dinamicamente as definições de tipos A

gerência das modificações feitas sobre as definições de tipos é mais complexa num ambiente de banco de dados do que nas linguagens de programação, devido ao fato dos dados num banco de dados serem persistentes e compartilhados. Além de não ser capaz de analisar o desempenho da operação de gerência de modificações, o benchmark de "Wisconsin" não é capaz de analisar o desempenho de outras operações como gerência de versões e controle de concorrência.

Existem outros benchmarks na literatura que podem ser utilizados para analisar o desempenho de SGBD's. Em [Anon88, Borr88, Tand88], é apresentado o benchmark "NonStop SQL" para avaliar a produtividade do processamento de transações on-line, com suporte a dados distribuídos e multiprogramação. Foi utilizado em análises realizadas nos SGBD's SQL/DS, DB2, IDM e R*.

Entretanto, este benchmark não é indicado para analisar outros parâmetros como CPU, E/S e operações de consulta. Consequentemente, não pode ser utilizado para analisar o desempenho do SGBD como um todo. Deste modo, não se aplica aos objetivos desta tese.

Em [Turb87, Tirb89], é proposto o benchmark "A83AP - ANSI SQL Standard Scalable and Portable" para avaliar o processamento realizado por SGBD relacionais que utilizam tipos de dados e funções no padrão ANSI SQL. Todavia, este benchmark teve como base o benchmark de "Wisconsin", sendo, também, sintético e utilizando dados e consultas sintéticos. Além dos parâmetros da

comparação normalmente encontrados em um benchmark, métodos de acesso e otimizadores de consulta são analisados em termos de desempenho. Para tanto, a comparação de sistemas é feita mediante proporções de equivalência de bancos de dados ("equivalent database ratio"), comparando-se tempos de resposta de consultas equivalentes em bancos de dados proporcionais, isto é, comparando-se parâmetros estáticos de SGBD's diferentes que possuem desempenho equivalentes.

Entretanto, os resultados gerados pelo benchmark não retratam com exatidão o desempenho real dos SGBD's. Isto se deve ao fato destes resultados serem sintéticos, baseados em suposições que nem sempre são satisfeitas pelo ambiente em uso. Deste modo, não será utilizado nesta tese que testará aplicações reais nos SGBD's que serão analisados.

Em [Edel90], é apresentado o benchmark "TP1", projetado para avaliar o desempenho de redes que possuem vários terminais de entrada de dados, realizam poucas transações de atualização nos dados e utilizam bancos de dados de médio porte. Todavia, este benchmark não suporta transações complexas, vários acessos ao banco de dados, junções, consultas baseadas em campos que não sejam chave primária única e bancos de dados de grande porte, e ignora os métodos de acesso usados pelo SGBD.

Logo, este berichmark rião pode ser. utilizado para avaliar o desempenho de qualquer. SGBD, Sua utilização deperide doi; objetivoa e caraterísPicaia das aplicações a serem

desenvolvidas na instalação e do próprio ambiente em uso. Portanto, seus resultados de desempenho são limitados, pelo fato de ignorar várias caracteristicas importantes de aplicações reais de banco de dados. Deste modo, também, não será utilizado nesta tese que testará aplicações reais.

Assim, algumas vezes pode não ser adequado utilizar um benchmark pronto. Esta decisão depende dos objetivos envolvidos na análise de desempenho.

b. Construir um novo benchmark

Consiste em construir um benchmark que atenda as necessidades e objetivos do ambiente em uso. Este benchmark deve ser capaz de testar todos os aspectos de um SGBD envolvidos na análise de desempenho em questão, e gerar parâmetros de comparação que permitam identificar este desempenho.

Geralmente, este benchmark gera reclamações por parte de vendedores cujos produtos não obtiveram desempenho satisfatório quando analisados por este benchmark.

O projeto e construção de um benchmark é uma tarefa complexa que requer um esforço considerável para ser implementada. Especificações incorretas podem gerar um benchmark ruim que, dificilmente, será utilizado na avaliação do desempenho de algum SGBD. Logo, para desenvolvê-lo, é necessário observar com cuidado todos os pontos que devem ser analisados e todos os fatores que influenciam na análise.

Para viabilizar este desenvolvimento, torna-se necessário usar-se uma metodologia de desenvolvimento de benchmarks, de forma a disciplinar o processo de desenvolvimento. Existem várias metodologias para desenvolver benchmarks. Posteriormente, no item III.5, será apresentada uma metodologia para escolha e desenvolvimento de benchmarks. Esta teve como base aquela proposta em [Ston85, Yao87].

c. Utilizar resultados fornecidos por vendedores

Consiste em utilizar os resultados de desempenho de SGBD's fornecidos por vendedores. Entretanto, estes resultados não são totalmente confiáveis, pois podem ser arbitrários e tendenciosos, precisando ser analisados com cuidado.

Vendedores com experiência em benchmarks são capazes de escolher um benchmark tendencioso que gere resultados de desempenho próximos do desejado. Isto se deve ao fato de vendedores não fornecerem resultados que constatem o desempenho inferior de seus produtos. Este fato prejudicaria as vendas e a imagem do produto.

Deve-se acrescentar que um vendedor pode testar uma versão mais avançada e ainda não disponível do produto, e fornecer estes resultados ao usuário como se fosse da última versão disponível [Ston85]. Assim, torna-se impossível reproduzir estes resultados no ambiente do usuário, utilizando-se versões distintas.

Vendedores podem, também, modificar o benchmark para aumentar o desempenho. Este torna-se impreciso em alguns pontos, permitindo a escolha de uma aplicação que executa mais rapidamente no ambiente do usuário.

Logo, deve-se tentar forçar os vendedores a gerar os resultados de desempenho levando em consideração as caracteristicas do ambiente operacional e das aplicações envolvidas no ambiente no qual será instalado o SGBD. Entretanto, o usuário recebe os resultados, mas também não é capaz de identificar as anomalias do SGBD, pois este foi testado no ambiente do vendedor sem a presença do usuário.

Em alguns casos, é possível testar o SGBD e o benchmark no ambiente do vendedor em presença do usuário. Esta solução não elimina as táticas utilizadas para manipular os resultados.

A solução é testar o SGBD e o benchmark no ambiente do usuário. Consegue-se identificar, imediatamente, anomalias no desempenho do SGBD e garante-se a seleção daquele cujo desempenho não sofre influência da esperteza de vendedores.

d. Testar o SGBD em aplicações reais

Consiste em testar o SGBD usando aplicações reais da instalação, e observar os resultados durante um período de tempo. Assim, consegue-se saber o comportamento dos problemas da instalação, e o funcionamento exato e as anomalias de cada SGBD no ambiente em uso. Por ser um

ambiente real e restrito, os dados obtidos podem ser reproduzidos a qualquer momento.

Deve-se contabilizar tempo e recursos através do uso de um pequeno benchmark. Não deve-se fazer uso de benchmarks complexos, pois estes irão requerer uma equipe para programá-lo e testá-lo no ambiente em uso.

Deve-se, também, testar funcionalidade, facilidade de instalação e utilização, segurança, facilidade de desenvolvimento de aplicações utilizando a linguagem de consulta disponível, completude da documentação, suporte técnico de representantes nacionais, dentre outros parâmetros estáticos que podem influenciar no funcionamento da instalação.

Testes de SGBD's em aplicações reais geram resultados não tendenciosos, sem arbitrariedades, e mais próximos da realidade do ambiente em uso.

Nesta tese, será utilizada uma combinação dos dois primeiros e da construção e utilização de benchmarks.

Independentemente da alternativa escolhida, os SGBD's e o benchmark devem ser testados, preferencialmente, no ambiente onde o produto irá funcionar após a homologação. Isto visa garantir que os resultados gerados representam o desempenho real do SGBD na instalação do usuário.

Os testes realizados em outro ambiente produzem resultados de desempenho que, dificilmente, podem ser reproduzidos no ambiente em uso. Isto se deve ao fato de ambientes

distintos diferirem nas técnicas de gerência de memória e de E/S, nas configurações de software/hardware e em outras características.

III.6 METODOLOGIA PARA ESCOLHA E DESENVOLVIMENTO DE BENCHMARKS

Para disciplinar o processo de escolha de benchmarks, deve-se utilizar uma metodologia que divida o processo em várias etapas. Em cada etapa, faz-se uso dos resultados da etapa anterior.

Segundo [YaoS87], uma metodologia de benchmark para SGBD's deve considerar as variáveis do sistema na avaliação do desempenho. Cada variável deve ser isolada para avaliar os efeitos desta sobre o sistema.

A metodologia apresentada em [Ston85, YaoS87] serviu de base para a construção da metodologia que será apresentada. Esta metodologia será composta de três fases, as quais serão descritas a seguir:

a. Projeto

Estabelece-se a configuração e as condições ambientais dos SGBD's a serem testados, a aplicação a ser usada e os dados de teste, os parâmetros de comparação a serem analisados e os testes a serem executados.

Após a definição dos parâmetros de comparação, estabelece-se uma forma real de medir tempo e recursos. Os parâmetros serão computados seguindo este critério.

O critério da medida usado pelo benchmark deve permitir a interpretação rápida, direta e sem ambigüidade dos resultados de desempenho. Geralmente, usa-se o relógio da máquina hospedeira para medir o tempo. A leitura do relógio ocorre imediatamente antes e depois da execução de cada evento que deve ser avaliado. Os resultados devem ser guardados num arquivo de saída ou anotados pelo analista de forma a evitar perda de informações.

Finalmente, decide-se qual a alternativa, dentre aquelas apresentadas no item III.4, que será adotada para o ambiente em questão. Devem ser consideradas todas as vantagens e desvantagens de cada alternativa, antes de ser tomada qualquer decisão.

b. Execução

Executa-se o benchmark e obtém-se os resultados de desempenho.

Inicialmente, deve-se executar o benchmark em modalidade monousuária, mesmo que o ambiente operacional em uso seja multiusuário [Bitt83, Bora84, DeWi85, Ston85].

O processo deve ser repetido várias vezes, de forma a confirmar os resultados de desempenho. Assim, garante-se os resultados gerados.

Finalmente, caso o ambiente operacional do usuário seja multiusuário, executa-se o benchmark em modalidade multiusuária e obtêm-se os resultados de desempenho dos SGBD's em condições reais de funcionamento. Deve-se, repetir o processo diversas vezes, de forma a confirmar

os resultados. Assim, reduz-se a influência de fatores como compartilhamento dos dados do banco de dados, da CPU e dos dispositivos de E/S, multiprogramação, concorrência, paralelismo, deadlock, e outros fatores externos que podem influenciar no desempenho do SGBD.

c. Análise

Analisa e compara os resultados de desempenho de cada SGBD. A análise deve ser feita com cuidado para não gerar conclusões imprecisas ou incorretas.

Em ambientes multiusuário, deve-se considerar a influência de fatores externos nos resultados de desempenho gerados pelo benchmark. Estes fatores podem fazer com que SGBD's tenham comportamentos diferentes nos dois ambientes.

Finalmente, escolhe-se o "melhor" SGBD para a instalação, com base nos parâmetros dinâmicos avaliados pelos benchmarks (desempenho) e nos parâmetros estáticos previamente analisados (funcionalidade).

Durante a análise, o usuário deve ter clareza dos objetivos do benchmark, dos parâmetros de comparação por ele utilizados e dos objetivos de cada um destes parâmetros. Desta forma, gera-se medidas de tempo e recursos que retratam os interesses e necessidades do usuário, e que podem ser utilizadas para comparar características reais de SGBD's e para mostrar o desempenho destes em relação a aplicação do usuário.

A construção, execução e avaliação de benchmarks em modalidade multiusuária é mais complexa que em modalidade monousuária. Deve-se analisar a influência dos fatores externos no cálculo dos resultados de desempenho.

SGBD's que não possuem desempenho satisfatório em ambientes monousuários, geralmente, também não o possuem em ambientes multiusuários [DeWi85]. Adicionalmente, usuários não aceitam SGBD's, cujo desempenho em ambientes multiusuário é significativamente inferior ao desempenho em ambiente monousuário [Ande89].

A execução de SGBD's em ambiente multiusuário não tem como meta prioritária a otimização de consultas de um usuário. As prioridades do sistema são a otimização do controle de concorrência, e do acesso e compartilhamento dos dados, a deteção de "deadlocks", a recuperação de falhas e o aumento do número de transações que podem ser executadas por segundo. A complexidade do ambiente não permite reproduzir, com exatidão, as atividades ocorridas durante um intervalo de tempo, não sendo possível regerar com precisão resultados de desempenho.

Logc, a c : :ı.:ão do benchmark em ambienıe moriousuário é mi i i iples, ırata E exeruta somente uma trarisação por vez, ideriıifica eom melhor prEisão oc; i*ecurcoc; requeridos e o tempo gaısto pclas aplicações, elimiria o tcmpo da espei-a pr. reruri oc; embora o ıempo dE E/S possa pei-mariecer inalterado e, os resultados de desempenho gerados podem sEr reproduzidos a qualquei- momErito e estão mais próximos da ra**a**lidade do SGBD.

Os resultados de desempenho obtidos da execução de um benchmark em ambiente monousuário, podem ser utilizados como base na interpretação de resultados gerados em ambiente multiusuários. Entretanto, a eliminação do paralelismo de tarefas e do compartilhamento da CPU em ambientes monousuário gera ociosidade de recursos (por exemplo, a utilização da CPU deixa os dispositivos de E/S ociosos e vice-versa), e a utilização destes resultados como base para a obtenção do desempenho do SGBD em ambiente multiusuário pode ser imprecisa.

Deve-se, também, considerar que alguns SGBD's possuem desempenho diferente em ambientes monousuário e multiusuário. Como causas podemos citar as técnicas de gerência de buffer utilizadas, os recursos do ambiente operacional consumidos, o sistema operacional utilizado, a sobrecarga da instalação, o paralelismo das tarefas e o compartilhamento dos recursos.

Obtêm-se o desempenho de SGBD's, independente do ambiente operacional utilizado, através da definição e análise de parâmetros que são implementados e utilizados pelo benchmark. Parâmetros de comparação contabilizam os resultados de desempenho obtidos durante a execução do benchmark. Geralmente, medem tempo e analisam os recursos envolvidos na execução do SGBD.

Foram definidos alguns parâmetros dinâmicos que serão utilizados pelos benchmarks para avaliar o desempenho dos SGBD's. A seguir, será apresentada uma descrição sucinta destes parâmetros.

a. Tempo de resposta

é o intervalo de tempo entre a submissão da aplicação e a apresentação dos resultados, isto é, o intervalo de tempo entre a chamada do SGBD e a apresentação dos resultados das consultas realizadas pela aplicação.

é influenciado pela complexidade da aplicação, volume da dados recuperados, características do ambiente operacional, número de usuários utilizando o SGBD (este último fator em ambiente multiusuário). Entretanto, segundo [Gios76], variações na sequência de chamadas ao banco de dados pela aplicação não possuem efeito significativo no tempo de resposta .

Segundo [Turb87, Turb89], o tempo de resposta não deve ser usado para determinar a eficiência com que um SGBD realiza operações relacionais. Isto se deve ao fato de haver diferenças entre tempos de resposta de consultas que usam operações iguais mas recuperam volumes de dados diferentes, e pelo fato deste parâmetro de comparação incluir a improdutividade no seu resultado.

b. Tempo de Execução

é o intervalo de tempo entre o início e o fim do processamento. é o somatório do tempo de CPU com o tempo de E/S. Não é considerado o tempo de espera por recursos em ambientes multiusuário.

é influenciado pela complexidade da aplicação e volume de dados a serem acessados.

Em sistemas monousuários, o tempo de execução coincide com o tempo de resposta. Em sistemas multiusuários, o compartilhamento de recursos aumenta o tempo de resposta de cada usuário, embora, o tempo de execução possa permanecer inalterado. Pesquisas [YaoS87] mostram que o tempo de resposta aumenta com o número de usuários usando a máquina hospedeira e o SGBD.

Mecanismos utilizados pelo SGBD para garantir a integridade e consistência dos dados em ambientes multiusuários, podem, também, aumentar o tempo de execução e, consequentemente, o tempo de resposta.

c. Tempo de CPU

É o tempo de processamento.

É influenciado pela quantidade de memória utilizada pela CPU, velocidade de processamento da máquina hospedeira e organização do espaço de armazenamento. A quantidade de memória é inversamente proporcional ao tempo gasto pela CPU para gerenciar a memória [Hawt85].

O tempo de CPU pode ser crítico em SGBD's que utilizam dispositivos de memória extendida e/ou técnicas de acesso/armazenamento para reduzir o tempo de E/S.

Existem vários métodos disponíveis para otimizar a utilização do CPU. Um método seria a distribuição do processamento da aplicação através do uso de uma máquina de banco de dados, conforme descrito, anteriormente, no item III.3.

Em ambientes multiusuário, um acréscimo no número de usuários que utilizam o SGBD aumenta o tempo de espera por fatias de tempo de CPU e diminui a fatia de tempo de CPU alocada a cada usuário [Fedo87].

d. Tempo de E/S

é o tempo de acesso aos dados situados no banco de dados.

é influenciado pelas formas de organização do espaço de armazenamento, pelas técnicas de acesso/armazenamento utilizadas pelos bancos de dados, pela quantidade de memória principal disponível para carregar os dados e pelos algoritmos utilizados para minimizar o número de acessos ao banco de dados.

e. Tempo de operação

é o intervalo de tempo entre o início e o fim do processamento de uma operação. Em [Mart86], o tempo de operação é definido como a divisão do número de instruções de processamento necessárias a execução da operação, pela velocidade de processamento da máquina hospedeira (exemplo: MIPS - instruções por segundo).

é influenciado pela complexidade da operação, o volume de dados envolvido e o número de relações referenciadas pela operação.

Deve-se construir um mecanismo (vários benchmarks de baixo nível) que execute cada operação em separado [Bitt83]. As consultas são isoladas e uma única consulta

é realizada por vez. Assim, garante-se que o tempo de operação obtido não sofreu influência de outras operações.

A velocidade com que SGBD's processam consultas depende da organização do espaço de armazenamento e técnicas de acesso/armazenamento utilizadas, organização das relações, estruturação da aplicação, dentre outros fatores. Segundo [Yao987], a operação mais complexa, cara e cnvolvendo maior tempo de processamento em um banco de dados relacional, é a junção,

f. Tempo de improdutividade ("overhead")

Tempo gasto pelo sistema (hardware e software) na execução de tarefas que não contribuem de forma líquida para seu funcionamento.

Envolve o tempo gasto pelo sistema para inicialização e carga do banco de dados, seleção dos caminhos de acesso, gerência dos buffers de memória, comunicação com o usuário, chamada e carga do benchmark, interpretação e validação das operações, deteção de erros, emissão de mensagens, apresentação de resultados de consultas, espera por recursos, dentre outras tarefas. Em [Cole87, Turb87], o tempo de improdutividade é calculado através da execução de uma "consulta vazia", isto é, sem comandos de CPU e E/S.

O tempo gasto para formatar e mostrar resultados de consultas na tela é relativamente alto [Bitt83]. Em alguns casos, é vantajoso escrever os resultados de consulta em disco, reduzindo o tempo de comunicação com

o usuário, mas aumentando o tempo de E/S e o tempo de improdutividade causado pela chamada ao sistema operacional para gravar dados em disco. Em outros casos, é vantajoso criar uma relação especial no banco de dados para guardar os resultados das consultas [Bitt88, Turb89]. Caso seja escolhida uma das duas últimas alternativas, o arquivo/relação já deve estar criado, para evitar aumentos na improdutividade, e deve apenas receber os resultados gerados pelas consultas.

Para reduzir o tempo de improdutividade, deve-se grupar consultas similares, isto é, aquelas que usam as mesmas relações e recursos, e executá-las sequencialmente [Bitt83]. Após executar todas as consultas da aplicação, obtém-se a improdutividade para cada grupo de consultas. Deve-se, então, calcular o tempo de improdutividade médio da aplicação através da média aritmética dos tempos de improdutividade obtidos da execução de cada grupo de consultas.

g. Produtividade ("Throughput")

é o volume de processamento realizado pelo SGBD num determinado intervalo de tempo. é a medida de eficiência de um SGBD em relação à quantidade de dados manipulados, quando comparada com a capacidade máxima de manipulação de dados do SGBD.

Geralmente, a produtividade é medida pelo número de operações ou transações realizadas por segundo.

A medida dos tempos, descritos anteriormente, depende do critério estabelecido na fase de projeto. Geralmente, utiliza-se o relógio da máquina hospedeira. Entretanto, independente do critério utilizado, deve-se guardar os resultados em um arquivo de saída ou anotá-los, de forma que possam ser acessados ao fim da análise de desempenho.

Uma forma de armazenar os resultados é definir uma relação que deve, comerite, ser utilizada para guardar os resultados de desempenho obtidos da análise. Entretanto, o desempenho global do SGBD poderia ser prejudicado em função dos acessos ao banco de dados que seriam necessários para guardar as informações de desempenho. Poderia haver um aumento no tempo da resposta e no tempo de E/S.

CAPÍTULO IV

SISTEMAS GERENCIADORES DE BANCO DE DADOS = SGBD's

Neste capítulo, serão apresentados aspectos referentes aos SGBD's existentes no mercado de microcomputadores, que foram selecionados e serão analisados pelos benchmarks.

Inicialmente, serão feitas considerações a respeito da escolha dos SGBD's e das decisões tomadas. Posteriormente, será apresentado um breve sumário sobre cada SGBD.

Finalmente, serão apresentadas considerações sobre os produtos, e um quadro da funcionalidade dos mesmos.

IV.1 ESCOLHA DOS SGBD's

Os benchmarks serão utilizados para testar alguns SGBD's existentes no mercado de microcomputadores. Há diversos produtos disponíveis no mercado que diferem entre si em objetivos, características, funcionalidade, portabilidade e desempenho. Foram escolhidos sete SGBD's relacionais. São eles: Copperel, dBase III/Plus, Dialog Plus/c, Oracle, Paradox, Rbase V, e Zim.

A escolha destes produtos resultou de pesquisas sobre SGBD's disponíveis no mercado de microcomputadores. Foram mantidos contatos com representantes nacionais que se mostraram interessados no uso destes na tese em questão.

Outros SGBD's existentes no mercado, também, foram analisados. Como exemplo podemos citar: Arco-Iris [Soua88, Tecn88, Tecn89a, Tecn89b], DataEase [Brow86b, Derf88, Dick86a, Ede189a, Info88a, Palm87, Petr87, Poor87, Poor89, Rama89a, Sand88, Sant87b, Seym88c, Soua88], Dataflex [Barb86, Chrs85, Corr88, Curo86, Derf88, Dick86b, Info88a, Info88c, Intr89a, Intr89b, Mars86, Petr87, Seym88c, Wong88], DBXL [Corr88, Petr87, Seym88c, Wong88], FoxBase [Derf88, Dick86a, Hart87b, PCMu88b, Petr87, Rube87, Schw88, Seym88c], Informix-SQL [Dick86a, Fink88, Fink90, Fran88, PCMu88a, Shaw88], Ingres-PC [Fink88, Fink90, Fran88, PCMu88a, PCMu88b, Shaw88], Knowledge-Man [Aaro86b, Derf88, Dick86a, Seym88c], Metafile [Bowm86b], Omnis Quartz [Chap88, Falk87, Seym88c], OS/2 [Ede189b], Q&A [Aaro86a, Bein88, Desp87, Kras86b, Seym88a, Whyt85], Revelation [Corr88, Curo86, Derf88, Dick86a, Phe186, Seym88c], Sensible-Solution [Dick86a, O'Ha84], SQL-Base [Fink88], Tas-Plus [Robe86], Unify [Curo86, Shaw88, Soua88], XDB [Dick86b, Ede189a, Fink88, Fink89, PCMu88a, PCMu88b, Shaw88], XQL [Ede189a, Fink88].

No apêndice B deste trabalho, são apresentadas características destes sistemas gerenciadores de banco de dados, que foram obtidas durante a análise dos mesmos para a seleção dos que seriam testados nesta tese. Entretanto, não foi possível a utilização dos produtos listados anteriormente, devido a fatores como: não existência de alguns no Brasil, falta de informação sobre sua representação nacional, falta de interesse de

representantes para a utilização de seus SGBD's na tese e não aplicabilidade ao ambiente operacional em questão.

Apesar da maioria ser classificado como relacional, alguns possuem modelos próprios, outros possuem características muito similares ao modelo em rede. Segundo [Codd85], a grande aceitação dos SGBD's relacionais no mercado de banco de dados pelas suas características e facilidades, fez com que produtos não relacionais passem apresentados como relacionais ou que algumas características sejam adicionadas aos mesmos de forma a aproximá-los dos relacionais mesmo que estes não atendam os requisitos mínimos necessários. Ainda segundo Codd, não existe um SGBD no mercado que seja completamente relacional. Logo, não deve-se escolhê-lo por quão relacional ele seja, mas pela possibilidade de ser extendido de forma a incorporar novas características, possuir suporte permanente de representantes nacionais e adaptar as aplicações da instalação as novas versões futuras com um mínimo de esforço.

Este trabalho considerou, somente, SGBD's disponíveis no mercado de microcomputadores para facilitar o acesso as informações e garantir seu funcionamento e utilização. Deve-se considerar, também, que serão utilizadas aplicações reais que retratam melhor a realidade quando executadas sob produtos já instalados em diversos ambientes.

Para realizar os testes, serão usadas versões específicas e compatíveis de SGBD's, pois os testes devem ser realizados dentro das mesmas condições ambientais. Entretanto, existem

no mercado, versões mais recentes e otimizadas de alguns SGBD's, que não serão utilizadas neste contexto. Portanto, os resultados obtidos, podem não refletir as características de desempenho da última versão de cada produto. Informações adicionais sobre a versão que foi utilizada neste trabalho, podem ser obtidas nas referências bibliográficas citadas na descrição de cada produto.

Serão usadas aplicações, recursos e ambiente operacional de interesse para este trabalho. O desempenho obtido será influenciado pela complexidade da aplicação, disponibilidade de recursos e ambiente operacional. Logo, os resultados de desempenho, obtidos através dos testes, poderão não retratar o desempenho exato dos SGBD's em outras condições ambientais.

Os resultados de desempenho gerados pelo benchmark precisam ser analisados. Para tanto, é necessário conhecer com precisão a funcionalidade de cada SGBD que será testado. Estudos realizados sobre os produtos, permitiram fazer uma descrição de suas principais características e capacidades, que são apresentadas no apêndice A, obedecendo os parâmetros estáticos fixados no capítulo III. São eles:

a. Estrutura física e organização do espaço de armazenamento
b. Tipos de dados
c. Dicionário de dados
d. Linguagem de definição e manipulação de dados
e. Interface com usuário
f. Integração com outros SGBD's

g. Qualidade da documentação

h. Segurança e proteção dos dados

i. Detecção de falhas e reconstrução do banca de dados

j. Concorrência

k. Ambiente operacional

l. Portabilidade e flexibilidade

m. Gerador de aplicações

n. Arquitetura

o. Utilitários

p. Outros

A seguir, será apresentado um breve resumo das principais características dos SGBD's. A fim de facilitar o acesso as informações sobre os SGBD's, a sequência em que serão apresentados é estritamente alfabética, não existindo, portanto, nenhuma outra conotação na sua precedência.

As informações que serão descritas a seguir, foram extraídas quase que inteiramente das referências bibliográficas citadas no apêndice A. Isto se deve ao fato destes trabalhos serem as únicas fontes de informação disponíveis para abordagem das características de cada SGBD.

IV.2 COPPEREL

é um SGBD baseado no modelo relacional clássico, desenvolvido na COPPE/UFRJ. Além das facilidades normalmente encontradas nos sistemas relacionais, o COPPEREL adiciona outras facilidades tais como: definição de asserções de integridade, procedimentos catalogáveis,

facilidade de auditoria e reconstrução, definição de modelos externos específicos utilizando relações virtuais, mecanismos para autorização de usuários, comandos interativos e condicionais, e uma estrutura física adequada para a implementação do modelo interno do banco de dados.

Quanto a **estrutura física e organização do espaço de armazenamento**, suas principais características são: armazenamento dos dados em tabelas, utilização da indexação por "hashing" para acessar/armazenar os dados, armazenamento dos dados de uma base de dados em um único arquivo físico, relocação automática de memória sem existência de espaço vazio entre tuplas, recursos para definição de vistas e organização do espaço de armazenamento por paginação.

Os **tipos de dados** suportados pelo COPPEREL são: caracter, real, inteira e natural. As funções do **dicionário de dados** são supridas por esquemas que armazenam informações e podem ser consultados por usuários através da linguagem de definição e manipulação de dados LOPEREL.

A **linguagem de definição e manipulação de dados** autocontida, denominada LOPEREL, dispensa o uso de uma linguagem hospedeira. Esta contém comandos para definição dinâmica, atualização, gerência e manipulação de bases de dados relacionais, controle de fluxo, expansão de procedimento, cálculo de expressão e comandos de E/S. Não existe, hoje, **interfaces com usuário** interativas por menus e gráficas. O SGBD pode ser utilizado interativamente ou por lotes e possui interface para linguagens hospedeiras.

R segurança e proteção dos dados é feita através da definição de senhas. A deteção de Falhas e reconstrução do banco de dados é feita pelo Subsistema de Reconstrução do SGBD COPPEREL (SR) que recupera falhas de transação, sistema e meio de armazenamento.

Quanto a concorrência, existe uma proposta de implementação multiusuária para o SGBD COPPEREL [Pale85]. Entretanto, esta não foi implementada. O sistema COPPEREL foi desenvolvido utilizando técnicas de portabilidade, de forma a permitir sua instalação em diferentes ambientes.

I SGBD i.tilizI uma arquitetura modu3ai- composta dE dois módulos priricipais: a máquinR virtual e o compi3ador da linguagem Loperal. O COPPEREL PC vem sando exploi-ado de forma a ser extendido à aplicações não convencionais e poder ser. aplicado a c3asses de apli±ar,õec, especificas e bariefi± iar vários projetos [Souz88].

IV.3 DBASE III/PLUS

é um SGBD relacional desenvolvido pela Ashton-Tate e, representado e comercializado no Brasil pela Datalógica.

Quanto a estrutura física e organização do espaço de armazenamento, suas principais características são: armazenamerito doE dados em tiEe1ac, u;o da indexaçÃo por Rrvore B+ para acessar/armazenar os dados e necEssidade de reorganizar. a memói-ia para liberar o espaço vazio eritra tup3as.

Os tipos de dados suportados pelo DBASE III PLUS são: caracter, numérico, lógico, memo e data. O catálogo funciona como um dicionário de dados não automatizado para uma base de dados.

A linguagem de definição e manipulação de dados interativa e isterpaEtada, denci inada dBASE, permite desenvolver programis, criir E maripular arqi ivos de dados e definii telas de E/S de dadrs. Uma interface cam iasuário isitEraEiva por menLs ermite consultaI e maniplII o banco de dados através da seleção de comandos.

A segurança e proteção dos dados é feita mediante definição de senhas e níveis de acesso ao banco de dados. Não há um mecanismo restaurador automático para deteção de Falhas e reconstrução do banco de dados.

Quanto a concorrência, existe um mecanismo de bloqueio que garante proteção, segurança e intégridade aos dados durante atualizações no banco de dados. O sistema DBASE III PLUS não foi desenvolvido utilizando técnicas de portabilidade.

IV.4 DIALOG PLUS/C

É um SGBD relacional desenvolvido, representado e comercializado no Brasil pela Soft Consultoria em Processamento de Dados Ltda.

Quanto a estrutura física e organização do espaço de armazenamento, suas principais características são: armazenamento dos dados em tabelas, uso da indexação por

árvore B+ para acessar/armazenar os dados e necessidade de reorganizar a memória para liberar o espaço vazio entre tuplas.

Os tipos de dados suportados pelo DIALOG PLUS/C são: caracter, numérico, lógico, memo, data, inteiro e tabela. O catálogo funciona como um **dicionário de dados** não automatizado para uma base de dados.

A linguagem de definição e manipulação de dados interativa e interpretada, denominada Dialog, permite desenvolver programas, criar e manipular arquivos de dados e definir telas de E/S de dados. Uma **interface com usuário** interativa por menus permite consultar e manipular o banco de dados através da seleção de comandos.

A segurança e proteção dos dados é feita mediante definição de senhas e níveis de acesso ao banco de dados. Não há um mecanismo restaurador automático para **deteção de Falhas** e reconstrução do banco de dados.

Quanto a **concorrência**, existe um mecanismo de bloqueio que garante proteção, segurança e integridade aos dados durante atualizações no banco de dados. O sistema DIALOG PLUS/C foi desenvolvido utilizando técnicas de **portabilidade**, de forma a permitir sua instalação em diferentes ambientes.

IV.5 ORACLE

é um SGBD baseado no modelo relacional, desenvolvido pela Oracle Corporation, repvesentado e comercializado no Brasil

na área de microcomputadores pela Compucenter Informática Ltda. e na área de mini e mainframes pela Oracle do Brasil.

Quanto a **estrutura física e organização do espaço de armazenamento**, suas principais características são: armazenamento dos dados em tabelas, utilização da indexação por árvore B+ e concentração para acessar/armazenar os dados, armazenamento dos dados de uma base de dados em vários arquivos físicos, relocação automática de memória sem existência de espaço vazio entre tuplas e recursos para definição de vistas.

Os **tipos de dados** suportados pelo ORACLE são: caracter, numérico e data. As funções do **dicionário de dados** são supridas pelo CASE*DICTIONARY que armazena informações dos objetos e garante a integridade e consistência dos dados gerados em todas as fases do desenvolvimento de aplicações.

A **linguagem de definição e manipulação de dados**, denominada SQL*PLUS, garante a recuperação, manipulação, definição e segurança dos dados de um banco de dados relacional. Esta faz uso da linguagem relacional SQL no padrão IBM. Uma interface com usuário interativa por menus permite consultar e manipular o banco de dados através da seleção de comandos.

A **segurança e proteção dos dados** é feita através da definição de senhas e níveis de acesso aos dados. A **deteção de Falhas e reconstrução do banco de dados** é feita por um mecanismo restaurador que recupera falhas de transação, sistema e meio de armazenamento.

Quanto a concorrência, existe um mecanismo de bloqueio que garante proteção, segurança e integridade aos dados durante atualizações no banco de dados. O sistema ORACLE foi desenvolvido utilizando técnicas de portabilidade, de forma a permitir sua instalação em diferentes ambientes.

O SGBD utiliza uma arquitetura aberta (SQL*STAR) composta de dois módulos principais: a máquina de gerência de dados relacional (ORACLE KERNEL) e um conjunto de ferramentas.

IV.6 PARADOX

é um SGBD relacional desenvolvido pela Borland International e, representado e comercializado no Brasil pela Intercorp do Brasil Ltda.

Quanto a estrutura física e organização do espaço de armazenamento, suas principais características são: armazenamento dos dados em tabelas, utilização da identificação por chaves para acessar/armazenar os dados, armazenamento dos dados de uma base de dados em vários arquivos físicos, necessidade de reorganizar a memória para liberar o espaço vazio entre tuplas e recursos para definição de vistas.

Os tipos de dados suportados pelo PARADOX são: caracter, numérico, inteiro, data e monetário. As funções do dicionário de dados permitem armazenar informações dos objetos e garante a integridade e consistência dos dados gerados em todas as fases do desenvolvimento de aplicações,

A linguagem de definição e manipulação de dados, denominada PAL, permite desenvolver programas, criar e manipular arquivos de dados e definir telas de E/S de dados, apesar de ser limitado e com recursos inferiores aos do modo interativo. Uma interface com usuário alternativa permite consultar e manipular o banco de dados através da seleção de comandos.

A segurança e proteção dos dados é feita através da definição de senhas e níveis de acesso aos dados. Não há um mecanismo restaurador automático para detecção de Falhas e reconstrução do banco de dados.

Quanto a concorrência, existe um mecanismo de bloqueio que garante proteção, segurança e integridade aos dados durante atualizações no banco de dados. O sistema PARADOX foi desenvolvido utilizando técnicas de portabilidade, de forma a permitir sua instalação em diferentes ambientes.

IV.7 RBASE SYSTEM V

É um SGBD relacional desenvolvido pela Microsoft Corporation e, representado e comercializado no Brasil pela Compucenter Informática.

A estrutura física e organização do espaço de armazenamento, suas principais características são: armazenamento dos dados em tabelas, utilização da indexação por árvore B+ e identificação por chaves para acessar/armazenar os dados, armazenamento dos dados de uma base de dados em vários arquivos físicos, não associada de

reorganizar a memória para liberar o espaço vazio entre tuplas e recursos para definição de vistas.

Os tipos de dados suportados pelo RBASE SYSTEM V são: caracter, numérico, inteiro, data, hora e monetário. As funções do dicionário de dados permitem armazenar informações dos objetos.

A linguagem de definição e manipulação de dados, denominada Rbase/SQL, garante a recuperação, manipulação, definição e segurança dos dados de um banco de dados relacional. Esta faz uso de uma variação da linguagem relacional SQL/IBM. Uma interface com usuário interativa por menus permite consultar e manipular o banco de dados através da seleção de comandos.

A segurança e proteção dos dados é feita através da definição de senhas e níveis de acesso aos dados. Não há um mecanismo restaurador automático para deteção de Falhas e reconstrução do banco de dados.

Quanto a concorrência, existe um mecanismo de bloqueio que garante proteção, segurança e integridade aos dados durante atualizações no banco de dados. O sistema RBASE SYSTEM V não foi desenvolvido utilizando técnicas de portabilidade.

IV.8 ZIM

É um SGBD relacional desesivolvido pela Zasithe Isiformation do Casiadá c, repi-esen4tado e comercializado sio Brasil pela RCM Iiformátiia Ltda.

Quanto a **estrutura física e organização do espaço de armazenamento**, suas principais características são: armazenamento dos dados em tabelas, utilização da indexação por árvore B+ balanceada para acessar/armazenar os dados, armazenamento dos dados de uma base de dados em vários arquivos físicos, recursos para definição de vistas e organização do espaço de armazenamento por paginação.

Os **tipos de dados** suportados pelo RBASE SYSTEM V são: caracter, numérico, inteiro e data. As funções do **dicionário de dados** permitem armazenar informações dos objetos.

A **linguagem de definição e manipulação de dados**, denominada ZIM/ADL, garante a recuperação, manipulação, definição e segurança dos dados de um banco de dados relacional. Uma **interface com usuário** interativa por menus permite consultar e manipular o banco de dados através da seleção de comandos.

A **segurança e proteção dos dados** é feita através da definição de senhas e níveis de acesso aos dados. A **deteção de Falhas e reconstrução do banco de dados** é feita por um mecanismo restaurador que recupera falhas de transação, sistema e meio de armazenamento.

Quanto a **concorrência**, existe um mecanismo de bloqueio que garante proteção, segurança e integridade aos dados durante atualizações no banco de dados. O sistema RBASE SYSTEM V não foi desenvolvido utilizando técnicas de **portabilidade**.

IV.9 CONSIDERAÇÕES SOBRE OS PRODUTOS

Conforme descrito no capítulo III, o desempenho de SGBD's deve ser analisado usando parâmetros estáticos e dinâmicos. Os parâmetros estáticos são utilizados para avaliar a funcionalidade, enquanto os dinâmicos para medir tempo. Neste capítulo, foi apresentada uma descrição sucinta das principais características e capacidades de cada SGBD por parâmetro estático, de modo a avaliar sua funcionalidade.

A escolha dos SGBD's, o contato com fornecedores e representantes nacionais, a obtenção dos produtos, o uso dos mesmos e a análise da funcionalidade permitiram fazer algumas observações e tirar algumas conclusões, que serão apresentadas a seguir.

O suporte diário a SGBD's por representantes nacionais independe da nacionalidade do produto. É difícil afirmar algo a respeito da superioridade ou inferioridade de cada representação. Os problemas rotineiros são solucionados adequadamente nos produtos nacionais e estrangeiros, pois o representante conhece seu funcionamento e características.

Problemas, as vezes, ocorrem quando é necessário adaptar um SGBD às características da uma instalação, para melhor usar sua potencialidade, sem implicar em modificações na mesma. O produto precisa gerar desempenho satisfatório, suprir as necessidades do ambiente em uso e não perder sua funcionalidade. Neste caso, há diferenças entre produtos nacionais e estrangeiros.

Os produtos estrangeiros foram desenvolvidos em outros países. Os projetistas que o conceberam, conhecem detalhes fundamentais da concepção do SGBD que, frequentemente, são necessários para melhor adaptá-lo ao ambiente da empresa. Estes detalhes são, muitas vezes, desconhecidos por seus representantes nacionais, que passam a conhecê-los após receberem solicitações para suporte a problemas raros detectados no ambiente e modificações no produto.

Os produtos nacionais possuem suporte mais próximo e, frequentemente, seus representantes são os próprios fabricantes (ex: COPPEREL e DIALOG PLUS/C). Detalhes de concepção são, geralmente, do conhecimento destes. Assim, o suporte a produtos não nacionais, muitas vezes, é mais difícil e requer mais tempo, pois o representante precisa contactar o fabricante para obter a solução ideal. Contudo, podem existir SGBD's nacionais com problemas semelhantes.

A documentação dos produtos nacionais é em português, facilitando a compreensão e aprendizado destes pelos usuários. A utilização dos estrangeiros requer um mínimo de conhecimento de inglês, dificultando, em alguns casos, seu uso por usuários leigos neste idioma.

Quanto a substituição de versões, o tratamento é praticamente igual em ambos os casos. A troca das mesmas é feita pelos representantes nacionais dos produtos.

A seguir será apresentado um quadro da funcionalidade dos SGBD's, de forma a facilitar a análise e resumir aspectos de cálculo. A formação do quadro teve como base os parâmetros estáticos fixados no item III.3.1.

	COPPEREL	DBASEIII/PLUS	DIALOG PLUS/C	ORACLE
a.	Indexação por "Hashing"	Indexação por Árvore B+	Isidexação por Árvore B+	Isidexação por Árvore B+ Concentração
b.	Caract Numérico Inteiro Natural Data	Caracter Numérico Lógico Memo Data	Caracter Numérico Lógico Memo Data Inteiro Tabela	Caracter Numérico Data Binário
c.	Esquema (DD ativo)	Catálogo (DD restrito não ativo)	Catálogo (DD restrito não ativo)	DD ativo
d.	Loperel	dBase	Dialog	SQL*PLUS (Variação SQL)
e.	Interativa sem menus	Interativa por menus	Interativa por menus	Interativa por menus
f.	nenhuma	nenhuma	c/ DBASE	total c/SGBDs que usam SQL., parcial ou outro
g.	português	inglês	português	inglês
h.	senhas níveis acesso	senhas níveis acesso	senhas níveis acesso	senhas níveis acesso
i.	Journal Image AI e BI files	sem mecanismo restaurador	sem mecanismo restaurador	Journal Image hI e BI files
j.	Bloqueio	Bloqueio	Bloquaio	Bloqucio
k.	Mono a Multiusuário	Mono e Multiusuário	Mono e Multiusuário	Mosio a Multiusuário
l.	PC-XT, PC-AT, IBM, UNISYS, UNIX	PC-XT, PC-AT	PC-XT, PC-AT, Cobra, i.NIX	Todos os ambientes
m.	não possui	sim (GA)	sim (Dialog-Ger)	sim (EASY*SQL)

Figura IV.1 - QUADRO DA FUNCIONALIDADE DOS SGBD's

	PARADOX	RBASE SYSTEM V	ZIM
a.	Identificação por chaves	Indexação por Árvore B+	Indexação por Árvore B+
b.	Caracter Numérico Iriteiro Data Monetário	Caracter Numérico Iriteiro Data Hora Monetário	Caracter Numérico Iriteiro Data
c.	DD ativo	DD ativo	DD não ativo
d.	PAL	Rbase/SQL	ADL
e.	Interativa por menus	Interativa por menus	Interativa por menus
f.	p/DBASE	c/DBASE c/SGBDs que usam SQL	c/DBRSE c/SGBDs que usam SQL
g.	inglês	inglês	inglês
h.	senhas níveis acesso	senhas níveis acesso	senhas níveis acesso
i.	restauração parcial	sem mecanismo restaurador	Journal Image AI e BI files
j.	Bloqueio	Bloqueio	Bloqueio
k.	Mono e Multiusuário	Mono e Multiusuário	Mono e Multiusuário
l.	PC-XT PC-AT IBM, VAX	PC-XT PC-AT	PC-XT. PC-AT Cobra, UNIX, VAX, IBM
m.	sim (PPP)	sim (EXPRESS)	sim (ZIM/DA)

Figura IV.2 - QUADRO DA FUNCIONALIDADE DOS SGBD's (Cont.)

CAPÍTULO V

BENCHREL

Neste capítulo, serão apresentados aspectos referentes ao ambiente operacional e benchmarks que serão utilizados na análise do desempenho dos SGBD's.

Inicialmente, serão feitas considerações a respeito da escolha do ambiente operacional.

Em seguida, serão apresentados os benchmarks que analisarão o desempenho dos SGBD's em termos de características, funcionamento e aplicação utilizados em sua implementação.

Os benchmarks de baixo nível serão, então, apresentados e executados nos diversos SGBD's. Serão apresentados resultados de desempenho e considerações sobre os SGBD's.

Finalmente, será implementado e executado um benchmark de alto nível e, apresentada uma análise do desempenho dos SGBD's.

V.1 AMBIENTE OPERACIONAL

O benchmark será executado juntamente com os SGBD's, em um microcomputador do tipo IBM-PC/AT operando na modalidade monousuária.

O microcomputador utiliza um processador principal do tipo INTEL 80286 com uma porta serial e duas portas paralelas,

sem co-processador, e com capacidade interna de 32 bits; dispõe de memória principal de 640 Kbytes (distribuída da seguinte forma: 64 Kbytes usados pelo DOS e pelos programas residentes, e 576 Kbytes para os programas de aplicação), memória ativa de 2 Megabytes (distribuída da seguinte forma: 640 Kbytes de memória principal, 32 Kbytes de memória "display" e 1408 Kbytes de memória extendida), EPROM de 16 KB para firmware, duas unidades de discos flexíveis de 5 1/4" D/D com capacidade por disco de 360 Kb e 1.2 Mb, respectivamente, uma unidade de disco rígido Seagate modelo ST157N com capacidade de 50 Mb, CPU com velocidade de processamento de 12 Mega/Hertz, interface RLL ("Run Length Limited") de 7.5 Megabits/s, interleave 2:1, índice de CPU ("Computing Index" - CI) do Norton de 13.7, índice de disco ("Disk Index" - DI) do Norton de 2.5, índice de desempenho ("Performance Index" - PI) do Norton de 9.9, controladora Seagate modelo ST02 e sistema operacional MS/DOS versão 3.3. O DOS será configurado com os parâmetros "Files = 50" e "Buffers = 20". Esta otimização (utilização de 20 áreas de memória principal para armazenar dados temporariamente e minimizar o número de acessos a disco) no parâmetro "Buffers" será possível pelo fato da máquina utilizada possuir 2 Mb de memória RAM.

As limitações no escopo do ambiente considerado visa gerar conclusões válidas sobre o desempenho dos SGBD's analisados, e isolar os efeitos causados pelas configurações de hardware, características do sistema operacional e softwares utilizados.

Segundo [Bitt83, DeWi85, Ston85], a análise de desempenho de SGBD's em qualquer ambiente operacional, deve iniciar com a execução do benchmark e dos SGBD's em ambiente monousuário. Assim, são, imediatamente, identificadas anomalias no desempenho dos mesmos e, garante-se que os resultados não sofrem influência de fatores externos como compartilhamento dos dados do banco de dados, da CPU e dos dispositivos de E/S, paralelismo e concorrência. Consegue-se, também, identificar com melhor precisão os recursos requeridos e o tempo gasto pelas aplicações, e os resultados de desempenho dos produtos.

O processo de execução dos benchmarks será repetido 5, 10, 100 e 1000 vezes, de forma a confirmar os resultados de desempenho e gerar parâmetros mais precisos e corretos. A execução de cada benchmark 1000 vezes verificará se os parâmetros tendem a se estabilizar a medida que o número de execuções aumenta e, obterá um tempo de operação mais próximo do real e com um mínimo de influência dos outros parâmetros dinâmicos. A cada mudança de SGBD no teste do benchmark e a cada mudança de benchmark, será zerado o buffer de memória para evitar a propagação de sujeiras e a influência de outros fatores no teste corrente.

A entrada dos dados da aplicação e dos comandos dos benchmarks será feita em massa através da leitura de blocos de registros via arquivo externo, de forma a reduzir o tempo de improdutividade que seria gasto para selecionar opções em menus e, digitar e fornecer informações para os SGBD's via tela. O tempo de E/S gasto para ler dados da tela ou do disco tende a ser igual.

Os resultados das c : : realizadas sobre os dados pelos besichmerks, seiÉo Rrmazenados em relaçõas especiais previamente criadas no banco de dados. O mesmo ocorrerá com os resultados de desempesilia. Dcste modo, alimina~sa o tempo de E/S que seria dispendido ria formatação a apresentação dos rasultados de consultas e dasempenho na Pela.

As relações resultantes de operações relacionais serão analisadas para verificar a eliminação das tuplas duplicadas. Assim, obtêm~se a forma como o SGBD trata os dados, analisando~se formas de eliminar redundâncias e garantir a integridade e unicidade dos dados.

Não serão usados os parâmetros dinâmicos para analisar a funcionalidade de cada SGBD, isto é, o desempenho dos parâmetros estáticos. Conforme descrito anteriormente, serão executados benchmarks padrões cujos comandos foram previamente armazenados em arquivos, não podendo comparar~ se os tempos obtidos por estes, com aqueles que seriam obtidos caso as entradas de dados fossem feitas via tela, ou com aqueles gerados por otimizadores de consulta, ou com aqueles gerados por variações nos parâmetros estáticos.

Caso os benchmarks sEjam extesididoc; em futuYoc; 1i.abaltios para ambiantes iultiusuái io, estes deveiãr: coni ider.Ai a influênci de fatores externos, incorporar novos parâmetros disiâmitos, ragarar os testes e veasialiiar os i-esultados. Oi parâmetros estáticos precisarão ser observados e apiofuridadoi ii ob a ótic c mul2 iusuári4. Oi resu1tados de desempenho obtidos em ambiente monousuário, poderão ser usados como base sia isiter.pretação dos i-esultadi3ii.

Serão usados SGBD's relacionais, pois estes dispõem de facilidades como o uso do conceito de normalização, de estruturas simples e flexíveis, de linguagens não navegacionais e de relacionamentos entre registros com base em associações simbólicas, isto é, através de comparação de valores de campo, e a manipulação simultânea de um conjunto de registros ao invés de processá-los um a um.

Nestes sistemas, o usuário não precisa conhecer a estrutura física interna do banco de dados, se preocupando apenas com o que ele deseja que seja feito pela aplicação e ignorando o como será feito. O desenvolvimento, implementação e manutenção destas ficam facilitados e tornam-se mais rápidos. Assim, as aplicações ficam independentes da estrutura física interna do banco de dados, pois existe um nível de abstração entre a forma de apresentação dos dados e a forma de armazenamento no banco de dados. Modificações no esquema interno não implicam em modificações nos programas de aplicação existentes na instalação, entretanto, podem afetar a eficiência dos mesmos.

Adicionalmente, estes sistemas são simples e poderosos, e já foram usados, com sucesso, em vários estudos. Alguns destes SGBD's (exemplo: Oracle e Rbase System V na versão corrente, e Paradox e dBase na próxima versão) utilizam a linguagem relacional não procedural SQL [Dowg88, Fink87, Pasc89, Shaw88, Wong88] como forma de padronizar a criação e o acesso a bases de dados relacionais, e facilitar a importação/exportação de aplicações e informações armazenadas em diferentes ambientes operacionais e portes de computadores. Deste modo, é possível integrar bancos de

dados locais desenvolvidos em microcomputadores com aqueles desenvolvidos em mainframes e minicomputadores. Este ambiente integrado permite manter os dados centralizados no mainframe e acessados através do microcomputador, sem perder algumas características como independência de processamento e interface gráfica.

V.2 ESPECIFICAÇÃO E PROJETO DOS BENCHMARKS

A especificação, projeto e desenvolvimento dos benchmarks tomou como base a metodologia proposta no capítulo III.

Benchmarks devem usar aplicações reais em seus testes para melhor simular o funcionamento de SGBD's na instalação em uso e, gerar resultados de desempenho mais representativos e próximos da realidade. Entretanto, em ambientes com vários tipos de aplicações, a construção de benchmarks para cada tipo de aplicação é uma tarefa custosa. A solução adotada é a construção de uma aplicação única, simples e genérica que engloba os requisitos e objetivos daquelas encontradas na instalação, e o uso desta nos benchmarks para analisar os parâmetros de desempenho.

De forma a permitir o desenvolvimento de um benchmark representativo de ambientes reais de banco de dados, foram analisadas aplicações reais que utilizam banco de dados, desenvolvidas no ambiente operacional da Eletrobrás Centrais Elétricas Brasileiras S. A., e observadas as operações mais frequentes. Apesar dos SGBD's utilizados pela empresa não serem exatamente os mesmos que serão

testados nesta tese, as transações envolvidas em aplicações de banco de dados independem do SGBD em uso e empregam combinações de operações de forma a atender seus objetivos. Entretanto, a amostra obtida é restrita e dependente das aplicações analisadas, e contém um conjunto reduzido de transações, sendo representativa somente para os ambientes considerados, não devendo ser generalizada para todos os ambientes de banco de dados.

A simplicidade e restrição desta amostra visa reduzir a complexidade desta primeira versão do benchmark, eliminar o esforço necessário a obtenção de outras amostras de transações reais usadas em outras empresas, e viabilizar o seu desenvolvimento a curto prazo. Contudo, os objetivos de assimilação e experimentação dos principais conceitos envolvidos na análise de desempenho foram alcançados.

Em futuros trabalhos, poderiam ser desenvolvidas novas estatísticas baseadas em amostras maiores de transações reais em uso em vários empresas que usam SGBD's. A nova amostra deveria conter um conjunto vasto de transações, seria representativa de ambientes reais de banco de dados e permitiria o desenvolvimento de benchmarks de alto nível típicos mais completos que poderiam ser usados para analisar o desempenho de grande parte dos SGBD's em vários ambientes operacionais que utilizam diversos tipos de aplicações. Para tanto, deveriam ser contactadas diversas empresas, de forma a analisar suas instalações e aplicações, observar as transações, montar a estatística e identificar as transações típicas mais frequentes.

Para montar amostras significativas e gerar estatísticas válidas, é necessário estabelecer classes de ocorrências de informações, de forma a distribuir os valores encontrados por classe e apurar a frequência de cada uma dentro da amostra. A apuração de informações por classe permite gerar estatísticas e expor os resultados obtidos da apuração em forma de tabelas ou gráficos.

Para construir a amostra de transações, definiu-se classes de transações típicas, que serão apresentadas a seguir. Fixou-se uma transação típica Ti como uma combinação de operações sobre relações. Cada Ti usa relações como operandos e produz uma relação como resultado.

T1 = Seleção

T2 = Projeção

T3 = União

T4 = Interseção

T5 = Diferença

T6 = Divisão

T7 = Produto Cartesiano

T8 = Junção

T9 = T1 + T8 = Seleção + Junção

T10 = T5 + T3 = Subtração + União

T11 = T1 + T5 + T3 = Seleção + Subtração + União

T12 = T1 + T2 = Seleção + Projeção

T13 = T1 + T5 = Seleção + Subtração

T14 = T1 + T2 + T8 = Seleção + Projeção + Junção

T15 = T1 + T3 = Seleção + União

T16 = T14 + T10 = Seleção + Projeção + Junção + Subtração + União

T17 = T14 + T5 = Seleção + Projeção + Junção + Subtração

T18 = T14 + T3 = Seleção + Projeção + Junção + União

A escolha destas transações típicas deve-se ao fato de serem estas as transações mais utilizadas pelas aplicações da instalação analisada. Variações nos objetivos de uma aplicação acarretam variações nas transações típicas Ti usadas pela aplicação e na frequência de uso de cada Ti.

Uma análise detalhada das aplicações da Eletrobrás permitiu gerar uma distribuição de frequência para as transações Ti. A seguir, serão apresentados os resultados da apuração em forma de tabela (figuras V.1 e V.2) e gráfico (figura V.3). As figuras mostram a frequência das transações típicas, num total de 3.165 transações e 17 aplicações (P1 ... P17).

T/A	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
T1	23	103	2	50	80	54	91	9	18	20
T2	3	2	3	2	2	13	8	3	6	8
T3	10	44	4	83	20	45	141	33	81	21
T4	6	58	2	12	8	9	16	1	3	11
T5	2	9	4	0	7	1	3	3	0	2
T6	0	8	0	0	0	0	0	0	0	0
T7	0	0	8	0	0	0	0	0	0	0
T8	61	a	8	7	0	10	0	3	0	1
T9	2	2	8	5	2	2	0	3	0	1
T10	0	0	8	0	0	0	0	0	0	0
T11	21	8	8	0	20	3	11	37	0	4
T12	3	5	1	1	0	0	5	1	0	7
T13	1	3	1	0	20	3	1	5	0	1
T14	0	35	2	0	7	0	11	30	1	62
T15	3	11	63	13	0	1	16	0	1	2
T16	11	i	8	7	0	2	9	0	2	13
T17	7	5	2	0	3	0	11	0	0	1
T18	5	5	8	5	0	1	7	1	1	13
TOTAL	158	23%	21	185	169	144	330	129	123	167

Figuro V.1 - DISTRIBUIÇÃO DE FREQUÊNCIA DAS TRANSAÇÕES TÍPICAS POR APLICAÇÃO

T/A	P11	P12	P13	P14	P15	P16	P17	%	TOTAL
T1	15	63	121	31	189	46	48	30,4	963
T2	5	15	12	5	6	25	9	4,0	127
T3	73	40	32	2	2	3	17	20,6	651
T4	5	12	20	1	13	12	6	4,6	147
T5	3	6	2	0	1	4	2	1,6	49
T6	0	0	0	0	0	0	0	0,0	0
T7	0	0	0	0	0	0	0	0,0	0
T8	0	10	4	5	82	13	0	6,3	200
T9	0	1	9	0	4	0	1	1,0	32
T10	0	0	0	0	0	0	0	0,0	0
T11	0	21	41	0	3	10	17	6,2	196
T12	3	11	6	1	3	13	2	1,8	58
T13	0	6	15	0	21	3	1	2,6	81
T14	0	1	20	0	184	17	2	11,8	372
T15	0	63	4	4	12	3	7	4,4	140
T16	0	3	1	3	0	5	1	1,8	58
T17	0	6	10	0	1	0	1	1,4	43
T18	0	6	0	3	0	4	1	1,5	48
TOTAL	104	264	297	55	521	158	115	100	3165

Figura V.2 - DISTRIBUIÇÃO DE FREQUÊNCIA DAS TRANSAÇÕES TÍPICAS POR APLICAÇÃO (Cont.)

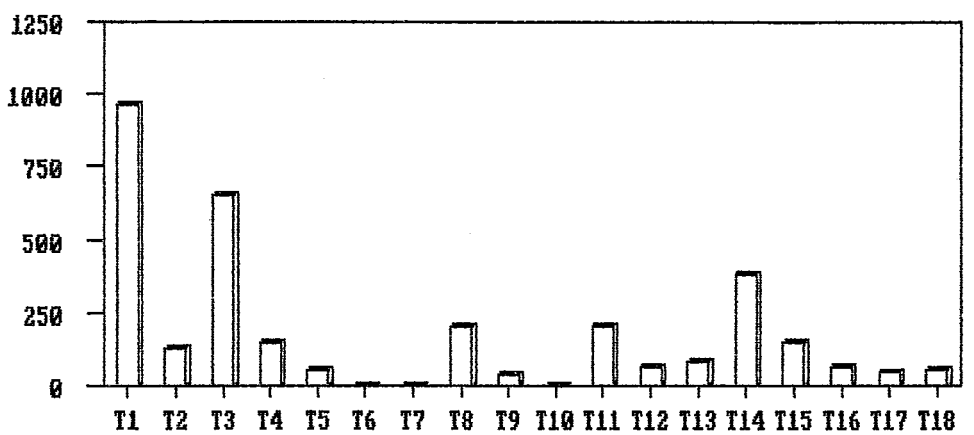

Figura V.3 - DISTRIBUIÇÃO DE FREQUÊNCIA DAS TRANSAÇÕES NA ELETROBRÁS

Os resultados obtidos permitiram concluir que as transações T1, T3 e T14 foram as mais freqüentes, com percentagens de 30.4%, 20.6% e 11.8%, respectivamente.

Tornou-se necessário, então, definir uma aplicação para fazer parte do benchmark, que usasse todas as transações típicas definidas anteriormente, com distribuição de frequência igual a apresentada na amostra. A solução encontrada foi utilizar uma aplicação da própria Eletrobrás, pois esta estaria projetada e especificada no mesmo ambiente operacional e seria representativa da amostra obtida. Entretanto, suas aplicações são muito extensas e um benchmark requer uma aplicação pequena, simples e concisa que consiga realizar todos os testes de desempenho necessários a avaliação. Logo, decidiu-se trabalhar com uma subparte de uma aplicação que atendesse os objetivos e requisitos obrigatórios.

A aplicação escolhida para ser usada pelos benchmarks para avaliar o desempenho dos SGBD's foi a AEI (Aquisição de Equipamentos de Informática). Esta baseia-se no mecanismo utilizado pela empresa para controlar a aquisição de equipamentos de informática pelos departamentos. É feito um acompanhamento da situação dos pedidos de equipamento e das tarefas executadas ao longo do processo de aquisição, podendo-se citar como principais etapas a atualização do cadastro de fornecedores, fabricantes, produtos e tipos de equipamentos, a especificação, preparação e aprovação do processo de aquisição, o empenho orçamentário e a compra dos equipamentos. A execução de todos os passos gera dados que são armazenados nos arquivos correspondentes.

A seguir, será apresentada uma simplificação (subparte) do diagrama entidade-relacionamento (E-R) da aplicação AEI, de forma a mostrar o seu funcionamento. A subparte escolhida é aquela estritamente necessária a demonstração dos conceitos aqui abordados. A íntegra da aplicação pode ser obtida nos manuais da mesma que estão disponíveis na empresa.

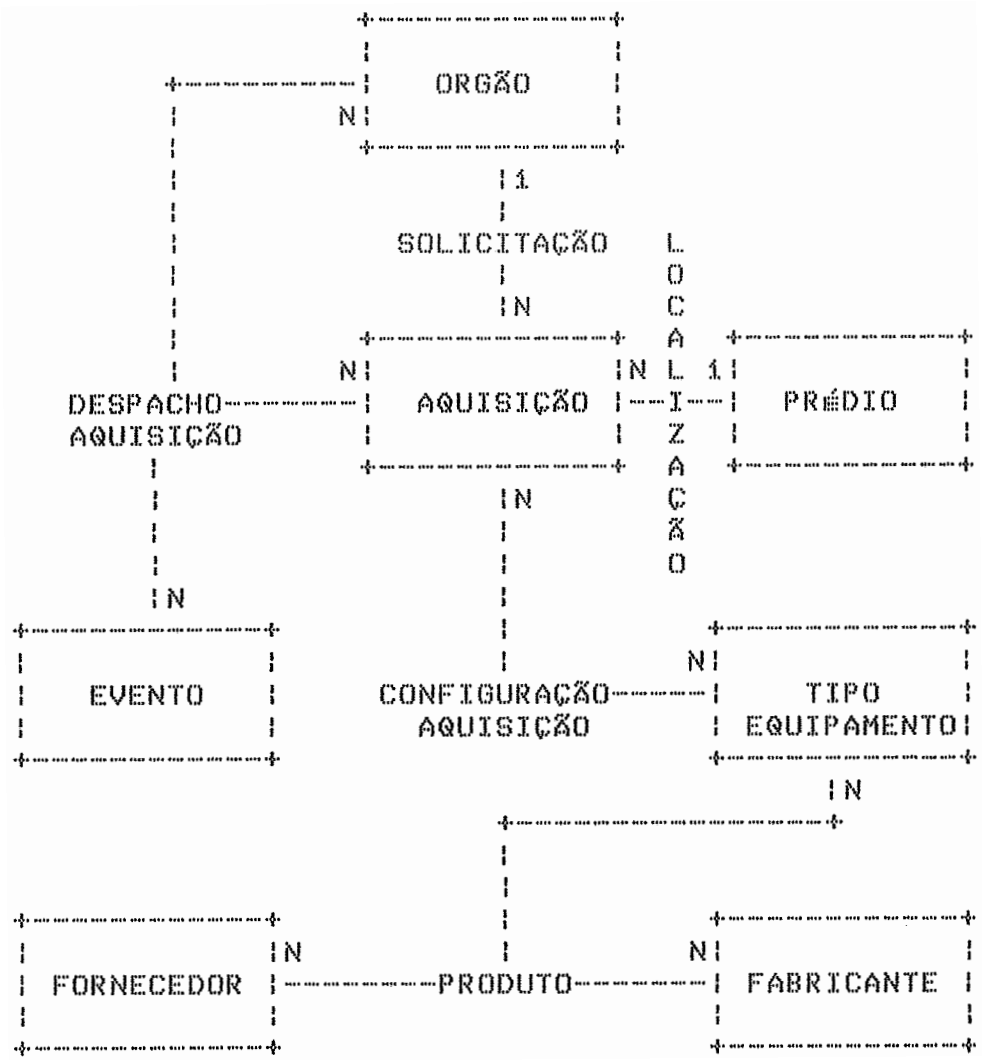

Figura V.4 – DIAGRAMA ENTIDADE–RELACIONAMENTO DA APLICAÇÃO AEI

Neste diagrama, foram suprimidos atributos e chaves das entidades e relacionamentos para não sobrecarregá-lo. No

apêndice D, são mostrados o modelo relacional normalizado correspondente ao diagrama E-R, e a massa de dados a ser usada na aplicação. Assim, pode-se analisar as relações, atributos e chaves que compõem a base de dados AEI.

As transações típicas foram implementadas na linguagem SQL usando as entidades da aplicação AEI. Foi usada a SQL na codificação, pois esta é uma linguagem relacional simples, padrão e de fácil entendimento. Usuários podem compreender melhor os benchmarks e, assim, as aplicações pelo fato desses estarem mais próximos do significado das transações. A descrição da implementação das transações típicas T1 ... T18 em SQL será mostrada a seguir.

a. T1a = Seleção por Valor (Obtenção de todos os detalhes dos fabricantes com sigla igual a "elebra")

```
SELECT * FROM fabricante
WHERE sigla-fabricante = "elebra";
```

b. T1b = Seleção por intervalo de valores (Obtenção de todos os dotal es dos fabricantes com sigla entre "compu" e "soft"]

```
SELECT * FROM fabricante
WHERE sigla-fabricante BETWEEN "compu" AND "soft";
```

c. T2 = Projeção (Obtenção da sigla de fabricantes de equipamentos)

```
SELECT sigla-fabricante FROM fabricante;
```

d. T3 = União (Obtenção dos fornecedores ou fabricantes de equipamentos)

```
SELECT    FROM fornecedor
UNION
SELECT * FROM fabricante;
```

e. T4 = Interseção (Obtenção doa fornecedores que são fabricantes de equipamentos)

```
SELECT * FROM fornecedor
INTERSECT
SELECT * FROM fabricante;
```

f. T5 = Diferença (Obtenção de fornecedores que não são fabricantes de equipamentos)

```
SELECT * FROM fornecedor
MINUS
SELECT * FROM fabricante;
```

g. T6 = Divisão (Obtenção das fornecedores que fornecem todos os tipos da equipamentos) PRODUTO DIVIDE BY TIPO-EQUIPAMENTO)

```
SELECT sigla-fornecedor FROM produto
GROUP BY sigla-fornecedor HAVING COUNT(*) = 34;
```

h. T7 = Produto Cartesiano (Obtenção dos produtos que poderiam existir caso todos as fabricantes fabricassem todos os tipos de equipamentos)

```
SELECT * FROM fabricante, tipo-equipamento;
```

i, T8 = Junção (Obtenção da descrição e dos detalhes dos equipamentos que são produtos)

```
SELECT * FROM produto, tipo-equipamento
WHERE produto.sigla-tipo-equipamento =
      tipo-equipamento.sigla-tipo-equipamento;
```

j. T9 = Seleção + Junção (Obtenção dos dados das aquisições de equipamentos solicitadas pelo "DEOI")

```
SELECT * FROM aquisição, configuração-aquisição
WHERE aquisição.sigla-orgão-solicitante = "DEOI"
AND configuração-aquisição.num-aquisição
     = aquisição.num-aquisição);
```

k. T10 = Subtração + União (Atualização da data da última compra de todos os produtos através da subtração de 1 ano em cada data, em virtude de erra administrativa)

```
UPDATE produto
SET data-última-compra = data-última-compra - 10000;
```

l. T11 = Seleção + Subtração + União (Substituição do nome do fornecedor cuja sigla é "veiga som" por "veiga som eletronicos ltda")

```
UPDATE fornecedor
SET nome-fornecedor = "veiga som eletronicos ltda"
WHERE sigla-fornecedor = "veiga som";
```

m. T12 = Seleção + Projeção (Obtenção da sigla dos equipamentos fornecidos pela "compumicro")

```
SELECT sigla-tipo-equipamento FROM produto
WHERE sigla-fornecedor = "compumicro";
```

n. T13 = Seleção + Subtração (Eliminação das solicitações de aquisição do "DEEC")

```
DELETE FROM aquisição
WHERE sigla-orgão-solicitante = "DEEC";
```

o. T14 = Seleção + Projeção + Junção (Obtenção da sigla dos equipamentos solicitados pelo "DERE")

```
SELECT configuração-aquisição.sigla-tipo-equipamento
FROM aquisição, configuração-aquisição
WHERE aquisição.sigla-orgão-solicitante = "DERE"
AND configuração-aquisição.num-aquisição
    = aquisição.num-aquisição);
```

p. T15 = Seleção + União (Obtenção dos detalhes dos fabricantes ou fornecedores com sigla maior ou igual a "micro")

```
SELECT * FROM fornecedor
WHERE sigla-fornecedor >= "micro"
UNION
SELECT * FROM fabricante
WHERE sigla-fabricante >= "micro";
```

q. T16 = Seleção + Projeção + Junção + Subtração + União (Atualização da quantidade de equipamentos solicitados pelo "deoi" com número do memo > 106500, acrescentando um equipamento aos pedidos)

```
UPDATE configuração-aquisição
SET quantidade-configuração=quantidade-configuração + 1
WHERE num-aquisição
IN (SELECT num-aquisição FROM aquisição
    WHERE sigla-orgão-solicitante = "deoi"
    AND num-memo-solicitação > 106500);
```

r. T17 = Seleção + Projeção + Junção + Subtração (Obtenção dos detalhes dos fabricantes que não fabricam o equipamento "mc01")

```
SELECT * FROM fabricante
MINUS
(SELECT sigla-fabricante, nome-fabricante
 FROM produto, fabricante
 WHERE produto.sigla-tipo-equipamento= "mc01"
 AND fabricante.sigla-fabricante =
     produto.sigla-fabricante);
```

s. T18 = Seleção + Projeção + Junção + União (Todo fabricante de equipamento "mc01" deve fornecê-lo. Atualizar o cadastro de fornecedores, acrescentando estes fabricantes)

```
SELECT sigla-fabricante, nome-fabricante
FROM produto, fabricante
WHERE produto.sigla-tipo-equipamento = "mc01"
AND fabricante.sigla-fabricante =
    produto.sigla-fabricante
UNION
SELECT * FROM fornecedor;
```

Cada transação típica Ti associada aos parâmetros de comparação (dinâmicos) previamente estabelecidos no capítulo III, originou um benchmark de baixo nível, usado para avaliar estes parâmetros. Tomou-se como base objetos (tabelas) e dados da aplicação AEI, sobre os quais

realizou-se as transações. No apêndice C, podem ser encontrados estes benchmarks implementados nas linguagens de definição e manipulação de dados usadas pelos SGBD's.

Após a construção dos benchmarks de baixo nível, foi definido um benchmark de alto nível utilizando a aplicação genérica AEI que englobou as transações mais freqüentes em um único programa. Este seguiu a distribuição de freqüência obtida anteriormente, obedeceu os requisitos e objetivos da aplicação, e avaliou os parâmetros de comparação.

As transações executadas pelo benchmark de alto nível corresponderam a combinação das transações executadas pelos benchmarks de baixo nível, obedecendo os mesmos critérios de busca e a mesma freqüência de utilização da amostra. Com isso, permitiu-se, também, a comparação dos resultados gerados pelos benchmarks de baixo e alto nível. Isto se deve ao fato do somatório dos resultados de baixo nível para cada parâmetro de comparação tender a ser próximo do resultado de alto nível gerado para cada parâmetro.

A implementação e execução dos benctimarks em cada SGBD utilizou os recursos E características comuns em todos os produtos. Portanto, não foram usadas técnicas espaciais r otimizar o desempenho, pois estas deveriam estar presentes Em todos os produtos a serem analisados, de maneira uniforme. Como exemplo podemos citar que nem todas as linguagens dispõem de compilador e aquelas que o possuem não fizeram uso nesta tese, de forma a manter o mesmo ambiente em todos os SGBD's e gerar resultados de desempenho comparáveis entre si.

Usou-se o mínimo de segurança durante a implantação dos testes de desempenho, para evitar que os tempos gerados pelos benchmarks fossem influenciados pelos mesmos. Mecanismos de segurança realizam teste a cada tentativa de execução de comandos e de acesso ao SGBD, banco de dados, arquivos e campos, para verificar o perfil de segurança do usuário e obter o nível de proteção previamente estabelecido. Portanto, o tempo de improdutividade gerado por estas operações é significativo e influencia no desempenho do SGBD. Consequentemente, os tempos obtidos da execução dos benchmarks, também, seriam influenciados.

V.3 BENCHMARKS DE BAIXO NÍVEL PROPOSTOS

Os parâmetros dinâmicos fixados no capítulo III, precisavam ser incorporados as transações típicas Ti de forma a gerar os benchmarks de baixo nível. Para tanto, precisou-se estabelecer uma forma real de medir tempo que permitisse a interpretação rápida, direta e sem ambiguidades dos resultados. A solução apresentada na metodologia, propôs a utilização do relógio da máquina hospedeira. Deveria-se medir o tempo imediatamente antes e depois da execução de cada evento a ser analisado.

Entretanto, precisava-se de um utilitário para marcar o tempo, realizar a leitura do relógio da máquina nos intervalos dos eventos. Decidiu-se utilizar o módulo NI ("Norton Integrator") do "Norton Utilities". Nesta, existam comandos que ativam a desativam o contador de tempo quando acionados ("TM START" e "TM STOP", respectivamente). Estes

comandos foram inseridos num arquivo auto-executável (.BAT) que ativa o SGBD e o benchmark a serem executados.

Cada transação típica Ti implementada em cada SGBD, foi executada utilizando o contador de tempo ("time") do Norton para contabilizar os parâmetros dinâmicos. O contador foi ativado imediatamente antes e depois da execução de cada evento, isto é, antes e depois da chamada do SGBD e benchmark (ex: TM START; COPPEREL BENCHMARK; TM STOP;)

A cada execução de um benchmark em um SGBD, foram analisados todos os parâmetros dinâmicos estabelecidos na metodologia. Entretanto, neste ambiente operacional deve-se fazer considerações sobre alguns destes parâmetros:

a. O **tempo_de_execução** coincidiu com o tempo de resposta. Isto se deve ao fato deste ambiente ser monousuário e não existir tempo de espera por recursos.

b. Durante a análise do tempo de execução, não foi possível distinguir o tempo de CPU do tempo de E/S. Seria necessário a construção de um procedimento que interceptasse os acessos a disco para medir a E/S.

c. O **tempo_de_improdutividade** retratou o intervalo de tempo entre a chamada do SGBD para inicialização e carga do banco de dados, e o término da execução dos procedimentos necessários a construção do ambiente para teste dos benchmarks. Para tanto, foi executada uma consulta vazia, isto é, sem comandos de CPU e E/S, de forma a melhor retratar todos os tempos improdutivos envolvidos na execução de uma aplicação.

Não foram medidos tempos de comunicação com o usuário, deteção de erros, emissão de mensagens, apresentação de resultados de consultas, espera por recursos, dentre outros tempos improdutivos. Conforme foi especificado anteriormente, cada benchmark será executado em batch, não havendo algum tipo de comunicação com o SGBD via tela para entrada de dados e comandos.

d. Fixou-se o **tempo_de_operação** (Toper) como o intervalo de tempo necessário para processar uma transação típica Ti. Logo, Toper coincidiu com o tempo de execução de cada benchmark de baixo nível (descontado o tempo de improdutividade), pois cada benchmark representa uma transação típica Ti.

e. Fixou-se a **produtividade** como o número de transações Ti realizadas por segundo. Logo, esta foi calculada pela fórmula 1/Toper (1 dividido pelo tempo de operação).

f. De forma a gerar parâmetros dinâmicos corretos e seguros, executou-se os benchmarks várias vezes. Para a maioria das transações i : Ti, a : i : que r .i entou-se r: número de execuções, o tempo total de execução (Texec) tendeu a aumentar uniformemente de t e o tempo médio de execução (Texec-médio) tendeu a um valor constante e próximo do Texec-médio anterior. Os tempos de execução que serão apresentados nos quadros a seguir (item V.4), são o Texec-médio.

g. A análise das aplicações da Eletrobrás gerou uma distribuição de frequência para as transações típicas Ti, que foi ilustrada nas figuras *V.1* e *V.2* deste

capítulo. Precisa-se considerar a influência desta na produtividade de cada transação Ti. Assim, representou-se esta **frequência** (Freq) nos resultados de cada SGBD.

h. Seja **produtividade_esperada** (esperança), o produto da frequência pela produtividade de cada transação Ti em cada SGBD. Assim, definiu-se o parâmetro **ProdEsp** nos resultados de cada produto. A **produtividade_média esperada** é obtida pelo somatório do ProdEsp de cada Ti.

No apêndice E, são apresentados quadros dos tempos de operação obtidos através da variação no número de execuções de cada transação Ti. A seguir, serão apresentados os resultados de desempenho gerados pelos benchmarks.

V.4 APRESENTAÇÃO DOS RESULTADOS DE DESEMPENHO E CONSIDERAÇÕES SOBRE OS SGBD'S

Os benchmarks foram executados várias vezes de forma a gerar parâmetros dinâmicos corretos e seguros. Assim, aumentou-se a complexidade de cada transação (executou-se n vezes Ti) e analisou-se os resultados para cada SGBD. A partir destes resultados, construiu-se quadros da distribuição dos tempos de operação das transações pelo número de execuções de Ti em cada SGBD, que serão apresentados no apêndice E. Conseguiu-se observar o comportamento de Ti em cada produto.

A implementação e execução dos benchmarks em cada SGBD gerou resultados de desempenho, e identificou deficiências. Consequentemente, existem diferenciações entre os tempos e

códigos gerados para cada produto. A seguir, serão apresentados os resultados obtidos dos benchmarks e algumas considerações sobre o desempenho dos SGBD's.

A fim de facilitar o acesso as informações, a sequência em que serão apresentadas é a mesma utilizada no capítulo IV para analisar a funcionalidade dos SGBD's, isto é, alfabética por produto.

Ti/Ti	Texec [2]	Toperação [2]	Produtividade [2]	Freq [2]	ProdEsp [2]
T1a	9,486	0,486	2,058	0,152	0,3128
T1b	9,807	0,807	1,239	0,152	0,1883
T2	9,878	0,878	1,139	0,040	0,0455
T3	11,317	2,317	0,432	0,206	0,0889
T4	9,745	0,745	1,342	0,046	0,0617
T5	11,325	2,325	0,430	0,016	0,0069
T6	11,985	2,985	0,335	0,000	0,0000
T7	10,792	1,792	0,558	0,000	0,0000
T8	10,593	1,593	0,628	0,063	0,0396
T9	15,973	6,973	0,143	0,010	0,0014
T10	9,801	0,801	1,248	0,000	0,0000
T11	9,801	0,801	1,248	0,062	0,0774
T12	10,371	1,371	0,729	0,018	0,0131
T13	9,403	0,403	2,481	0,026	0,0645
T14	15,981	6,981	0,143	0,118	0,0169
T15	11,527	2,527	0,396	0,044	0,0174
T16	16,912	7,912	0,126	0,018	0,0023
T17	12,417	3,417	0,293	0,014	0,0041
T18	13,678	4,678	0,214	0,015	0,0032
Total				1,000	1,4452

Obs:
1) Improdutividade = 9s
2) Definições nos itens a, d, e, g, h do item V.3

Figura V.5 - RESULTADOS DE DESEMPENHO DAS TRANSAÇÕES TÍPICAS NO COPPEREL (em segundos)

As considerações sobre a codificação, implementação e execução dos benchmarks no COPPEREL serão apresentadas no

capítulo VI. Por isso, estas não foram listadas após o quadro de desempenho, como o foram para os demais produtos.

Ti/T	Texec	Toperação	Produtividade	Freq	ProdEsp
T1a	3,143	0,143	6,993	0,152	1,0629
T1b	3,603	0,603	1,658	0,152	0,2520
T2	3,532	0,532	1,880	0,040	0,0752
T3	23,920	20,920	0,048	0,206	0,0099
T4	7,026	4,026	0,248	0,046	0,0114
T5	–	–	–	0,016	–
T6	–	–	–	0,000	–
T7	10,286	7,286	0,137	0,000	0,0000
T8	4,593	1,593	0,628	0,063	0,0396
T9	78,844	75,844	0,013	0,010	0,0001
T10	3,368	0,368	2,717	0,000	0,0000
T11	4,111	1,111	0,900	0,062	0,0558
T12	3,610	0,610	1,639	0,018	0,0295
T13	3,979	0,979	1,021	0,026	0,0265
T14	78,830	75,830	0,013	0,118	0,0015
T15	6,000	3,000	0,333	0,044	0,0146
T16	136,611	133,611	0,007	0,018	0,0001
T17	–	–	–	0,014	–
T18	6,251	3,251	0,308	0,015	0,0046
Total				1,000	1,5837

Obs:
1) Improdutividade = 3s
2) Definições nos itens a, d, e, g, h do item V.3

Figura V.6 – RESULTADOS DE DESEMPENHO DAS TRANSAÇÕES TÍPICAS NO DBASE III/PLUS (em segundos)

A implementação dos benchmarks no DBASE III/PLUS enfrentou algumas dificuldades por falta de recursos na linguagem. A seguir, serão apresentadas algumas considerações sobre os recursos disponíveis no SGBD.

i. Os comandos de manipulação de arquivos (COPY, JOIN, REPLACE, DELETE) obrigam a criação de um outro arquivo em disco para armazenar os registros resultantes da

operação. Deste modo, as operações relacionais tornam-se lenta e pouco eficiente, e não possuem os objetivos reais fixados na álgebra relacional. Uma das principais causas é a quantidade de E/S envolvida na criação e inserção de registros no novo arquivo. Adicionalmente, a execução destas requer espaço em disco, condicionando seu sucesso a existência de quantidade de memória secundária suficiente para o armazenamento do arquivo.

Deveria ser permitida a geração de arquivos temporários que seriam deletados ao fim de cada sessão DBASE, caso não fossem transformados em permanentes.

ii. Os comandos de busca (FIND, SEEK, LOCATE) não geram "loop", sendo necessário controlar as interações através do uso de comandos de laço. A última chave pesquisada no arquivo via busca, é usada para buscar a próxima chave que satisfaz a condição especificada. Assim, estes devem ser codificados entre cláusulas de início e fim de "loop". Consequentemente, a seleção não é automática, precisando ser controlada por programa pelo usuário, e aumentando o tempo de busca.

Outra alternativa é utilizar os comandos de cópia (COPY, APPEND FROM) para fazer a seleção. São especificadas cláusulas de seleção que informam quais os registros que devem ser copiados para o novo arquivo. Entretanto, esta alternativa possui os inconvenientes já citados, anteriormente, no item (i).

iii. Não há comandos que permitam leitura sequencial de arquivos (ex: READ ARQUIVO). Assim, deve-se simular o

"loop" da forma como foi descrito, anteriormente, no item (ii), condicionando seu término a uma condição. Consequentemente, aumenta-se o número de comandos a serem executados e o tempo de leitura do arquivo.

iv. Comandos de manipulação de arquivos (COPY, APPEND FROM, REPLACE) podem gerar registros repetidos no banco de dados. O DBASE III/PLUS dispõe de um parâmetro (SET UNIQUE) que quando ativado impede a duplicação de campo(s) chave(s) no arquivo de índice. Deste modo, comandos de gravação de registros cuja chave já exista no arquivo de índice, serão efetuados, somente, no banco de dados. Consequentemente, não evita-se a duplicação destes registros, que deve ser eliminada por programa. O controle sobre o conteúdo do banco de dados deve ser feito pelo usuário.

v. Comandos de manipulação de arquivos (APPEND FROM, JOIN, REPLACE, DELETE) obrigam a especificação de uma condição na sintaxe para gerar "loop". A omissão desta cláusula limita o processamento ao registro corrente. Arquivos posicionados no fim não são processados.

Isto se deve ao fato destes comandos não terem sido projetados com este objetivo, e não possuírem as características relacionais. Assim, a inclusão de condições abrangentes que não impessam a execução completa da operação, aumenta seu tempo de execução e torna o processamento mais lento.

vi. Não há como implementar as operações de diferença e divisão. A linguagem Dbase não é relacional, não

dispondo de recursos que suportem as operações básicas da álgebra relacional. Deste modo, seria necessário desenvolver procedimentos em linguagem de terceira geração que simulassem estes recursos.

Entretanto, estes procedimentos seriam lentos, e requisitariam maior programação, sendo compostos de vários comandos. Este não é o objetivo deste trabalho que visa analisar o desempenho de SGBD's através da execução de benchmarks construídos incorporando as transações relacionais mais frequentes de uma instalação. Assim, deve-se analisar o suporte do produto as transações Ti, a disponibilidade de comandos para realizar as operações relacionais básicas e o desempenho destes em relação ao de outros produtos.

vii. O comando de união de arquivos (APPEND FROM) não gera um arquivo em disco para armazenar os registros resultantes da operação. Os registros de um são armazenados no final do outro, sem eliminação dos repetidos. Deste modo, o conteúdo original é alterado e perde-se as informações previamente armazenadas.

Uma solução é gerar cópia do arquivo antes de realizar a operação. Assim, conserva-se os dados, permitindo manipulações futuras.

Em termos de E/S, a quantidade realizada pelo comando é praticamente a mesma daquela dos outros comandos que geram outro arquivo em disco. A criação/inserção de registros requer espaço em memória secundária para

armazená-los, condicionando seu sucesso a existência de espaço suficiente para seu armazenamento.

É permitida a união de arquivos com tipos de dados diferentes. Para tanto, apenas os campos existentes em ambos são copiados, não precisando que estes estejam na mesma ordem, mas que tenham o mesmo nome. Logo, os campos do arquivo fonte não comuns ficam sem informação e o arquivo resultante fica incompleto.

Entretanto, a obrigatoriedade dos campos comuns terem o mesmo nome prejudica a manipulação de arquivos com mesmo tipo de dados, mas nomes de campos diferentes. É necessário, então, gerar os arquivos com nomes de campos iguais de forma a permitir operações futuras.

viii. A operação de junção (JOIN) elimina a segunda ocorrência dos atributos comuns (mesmo nome) das duas entidades, independente destes possuírem mesmo domínio básico, conteúdo e tipo. As tuplas resultantes contêm, somente, a ocorrência destes atributos obtida da segunda entidade definida no comando. Atributos da primeira entidade com mesmo nome e diferentes tipos ou valores são eliminados do resultado.

Estas características contrariam as definições da álgebra relacional para junção natural. Isto, apenas, poderia ocorrer sobre atributos pertencentes ao mesmo domínio básico. Porém, pode-se especificar no comando os campos que devem aparecer na relação resultante, que não podem ter mesmo nome. Logo, cabe ao usuário decidir se o

atributo com mesma identificação será herdado da primeira ou segunda entidade.

Logo, o produto cartesiano entre dois arquivos que possuem atributos com mesmo nome, não funciona. Não existe comando próprio para esta operação. Deve-se usar o comando JOIN com a especificação de uma condição abrangente que não impeça a execução completa da operação. Porém, a eliminação da segunda ocorrência do atributo comum, ao invés de qualificá-lo como na álgebra relacional, impede sua execução com sucesso.

ix. A junção é a operação relacional mais lenta do DBASE III/PLUS. O não uso de índices, a leitura sequencial dos arquivos usados na transação e a obrigatoriedade da especificação de pelo menos uma condição no comando são algumas das causas da redução de velocidade.

x. A operação de união reduz seu desempenho a medida que aumentamos o número de execuções de cada transação ou o tamanho dos arquivos. A execução sequencial de uma transação por diversas vezes, aumenta muito um dos arquivos envolvidos na operação (vide vii) e, consequentemente, o tempo de execução da mesma. Neste trabalho, executou-se várias vezes uma transação de forma a gerar resultados mais seguros. Neste caso, o desempenho da união vai se prejudicando ao longo da execução, pois um dos arquivos vai crescendo a medida que a reexecutamos.

xi. Conforma descrito no item v.), a operação de junção obriga a especificação de uma condição, que igual a

sintaxe dos comandos DBASE das transações T8 e T9. Entretanto, a utilização de arquivos diferentes com massas de dados diferentes permitiu concluir que o tempo de execução de T9 é muito maior. Isto se deve a variação no número de registros de cada arquivo. Um aumento na massa de dados acarretou um aumento de 36 vezes no número de comparações e um aumento quase proporcional (27 vezes) no tempo de execução (vide benchmarks, massas de dados e tempos de resposta nos apêndices C, D, e E, respectivamente).

Ti/Tj	Texec	Toperação	Produtividade	Freq	ProdEsp
T1a	5,235	0,235	4,255	0,152	0,6468
Tib	5,468	0,468	2,137	0.552	0,3248
T2	5,487	0,487	2,053	0,040	0,0821
T3	20,171	15,171	0,066	Vi, 206	0,0136
T4	10,672	5,672	0,176	0,046	0,0081
T5	—	—	—	0,016	—
T6	—	—	—	0,888	—
T7	8,881	3,881	0,258	8, 000	0,0000
T8	6,600	1,600	0,625	0,063	0,0394
T9	58,018	53,018	0,019	0,010	0,0001
T10	5,266	0,266	3,759	0,000	0,0000
T55	5,811	0,811	1,233	0,062	0,0764
T12	5,600	0,600	1,667	0,018	0,0300
T13	5,641	0,641	1,560	0,026	0,0406
Ti4	58,018	53,018	0,019	0,118	0,0022
T55	6,400	1,400	0,714	0,044	0,0314
T16	76,999	71,999	0,014	0,018	0,0003
T17	—	—	—	0,014	—
T18	7,999	2,999	0,333	0,015	0.0049
Total				1,000	1,3007

Obs:
1) Improdutividade = 5s
2) Definições nos itens a, d, e, g, h do item V.3

Figura V.7 - RESULTADOS DE DESEMPENHO DAS TRANSAÇÕES TÍPICAS NO DIALOG PLUS/C (em segundos)

A implementação dos benchmarks no DIALOG PLUS/C enfrentou algumas dificuldades por falta de recursos na linguagem. A seguir, serão apresentadas algumas considerações sobre os recursos disponíveis no SGBD.

i. Os comandos de manipulação de arquivos (COPIE, JUNTE, SUBSTITUA, ELIMINE) obrigam a criação de um outro arquivo em disco para armazenar os registros resultantes. Assim, as operações relacionais tornam-se lenta e pouco eficiente, e não possuem os objetivos reais fixados na álgebra relacional. Uma das principais causas é a quantidade de E/S envolvida na criação e inserção de registros no novo arquivo. Adicionalmente, a execução destas requer espaço em disco, condicionando seu sucesso a existência de quantidade de memória secundária suficiente para o armazenamento do arquivo.

   Deveria ser permitido a geração de arquivos temporários que seriam deletados ao fim de cada sessão DIALOG, caso não fossem transformados em permanentes.

ii. Os comandos de busca (PROCURE, PESQUISE, LOCALIZE) não geram "loop", sendo necessário controlar as iterações através do uso de comandos de laço para simular o "loop". A última chave pesquisada no arquivo via busca, é usada para a próxima que satisfaz a condição especificada. Assim, estes devem ser codificados entre cláusulas de início e fim de "loop". Consequentemente, a seleção não é automática, precisando ser controlada por programa pelo usuário, e aumentando o tempo de busca.

Outra alternativa é utilizar os comandos de cópia (COPIE, ANEXE) para fazer a seleção. São especificadas cláusulas de seleção que informam quais os registros que devem ser copiados para o novo arquivo. Entretanto, esta alternativa possui os inconvenientes já citados, anteriormente, no item (i).

iii. Não há comandos que permitam leitura sequencial de arquivos (ex: READ ARQUIVO). Assim, deve-se simular o "loop" da forma como foi descrito, anteriormente, no item (ii), condicionando seu término a uma condição. Consequentemente, aumenta-se o número de comandos a serem executados e o tempo de leitura do arquivo.

iv. Comandos de manipulação de arquivos (COPIE, COMBINE, JUNTE, SUBSTITUA, ELIMINE) podem gerar registros repetidos no banco de dados. O DIALOG PLUS/C dispõe de um parâmetro (POE UNICO) que quando ativado impede a duplicação de campo(s) chave(s) no arquivo de índice. Deste modo, comandos de gravação de registros cuja chave já exista no arquivo de índice, serão efetuados, somente, no banco de dados. Consequentemente, não evita-se a duplicação destes registros, que deve ser eliminada por programa. O controle sobre o conteúdo do banco de dados deve ser feito pelo usuário.

v. Comando de manipulação de arquivo (COMBINE, JUNTE, SUBSTITUA, ELIMINE) obrigam a especificação de uma condição na sua sintaxe para gerar "loop". A omissão desta cláusula limita o processamento ao registro corrente. Arquivos posicionados no fim não são

processados. Deste modo, a operação relacional de UNIÃO que por definição não necessita de uma condição para unir dois arquivos, deve incorporá-la a sua especificação no comando COMBINE.

Isto se deve ao fato destes comandos não terem sido projetados com este objetivo, e não possuírem as características relacionais. Assim, a inclusão de condições abrangentes que não impessam a execução completa da operação de UNIÃO, aumenta seu tempo de execução e torna o processamento mais lento.

vi. Não há como implementar as operações de diferença e divisão. A linguagem Dialog não é relacional, não dispondo de recursos que suportem as operações básicas da álgebra relacional. Deste modo, seria necessário desenvolver procedimentos em linguagem de terceira geração que simulassem estes recursos.

Entretanto, estes procedimentos seriam lentos, e requisitariam maior programação, sendo compostos de vários comandos. Este não é o objetivo deste trabalho que visa analisar o desempenho de SGBD's através da execução de benchmarks construídos incorporando as transações relacionais mais frequentes de uma instalação. Assim, deve-se analisar o suporte do produto as transações Ti, a disponibilidade de comandos para realizar as operações relacionais básicas e o desempenho destes em relação ao de outros produtos.

vii. O comando de união de arquivos (ANEXE DE) não gera um arquivo em disco para armazenar os registros resultantes

da operação. Os registros de um são armazenados no final do outro, sem eliminação dos repetidos. Deste modo, o conteúdo original é alterado e perde-se as informações previamente armazenadas.

Uma solução é gerar cópia do arquivo antes de realizar a operação. Assim, conserva-se os dados, permitindo manipulações futuras.

Em termos de E/S, a quantidade realizada pelo comando é praticamente a mesma daquela dos outros comandos que geram outro arquivo em disco. A criação/inserção de registros requer espaço em memória secundária para armazená-los, condicionando seu sucesso a existência de espaço suficiente para seu armazenamento.

é permitida a união de arquivos com tipos de dados diferentes. Para tanto, apenas os campos existentes em ambos são copiados, não precisando que estes estejam na mesma ordem, mas que tenham o mesmo nome. Logo, os campos do arquivo fonte não comuns ficam sem informação e o arquivo resultante fica incompleto.

Entretanto, a obrigatoriedade dos campos comuns terem o mesmo nome prejudica a manipulação de arquivos com mesmo tipo de dados, mas nomes de campos diferentes. é necessário, então, gerar os arquivos com nomes de campos iguais de forma a permitir operações futuras.

viii. A operação da junção (JOIN) elimina a segunda ocorrência dos atributos comuns (mesmo nome) das duas entidades, independente destes possuírem mesmo domínio

básico, conteúdo e tipo. As tuplas resultantes contêm, somente, a ocorrência destes atributos obtida da segunda entidade definida no comando. Atributos da primeira entidade com mesmo nome e diferentes tipos ou valores são eliminados do resultado.

Estas características contrariam as definições da álgebra relacional para junção natural. Isto, apenas, poderia ocorrer sobre atributos pertencentes ao mesmo domínio básico. Entretanto, pode-se especificar no comando os campos que devem aparecer na relação resultante, que não podem ter mesmo nome. Logo, cabe ao usuário decidir se o atributo com mesma identificação será herdado da primeira ou segunda entidade.

Consequentemente, o produto cartesiano entre dois arquivos que possuem atributos com mesmo nome, não funciona. Não existe comando próprio para esta operação. Deve-se utilizar o comando JOIN com a especificação de uma condição abrangente que não impessa a execução completa da operação. Contudo, a eliminação da segunda ocorrência do atributo comum, ao invés de qualificá-lo conforme definição da álgebra relacional, impede sua execução com sucesso.

ix. A junção é a operação relacional mais lenta do DIALOG PLJ /C. A não utiliaação de índices, a leitura sequencial dos arquivos envolvidos à transação e a obrigatoriedade da especificação de pelo menos uma condição rici comando são algumas das causas da redução de

velocidade. A medida que a massa de dados aumenta, o tempo de resposta torna-se cada vez mais lento.

x. A operação de união reduz seu desempenho a medida que aumentamos o número de execuções da cada transação ou o tamanho dos arquivos. A execução sequencial de uma transação por diversas vezes, aumenta muito um dos arquivos envolvidos na operação (vide vii) e, consequentemente, o tempo de execução da mesma. Neste trabalho, executou-se várias vezes uma transação de forma a gerar resultados mais seguros. Neste caso, o desempenho da união vai se prejudicando ao longo da execução, pois um dos arquivos vai crescendo a medida que a reexecutamos.

xi. Conforme descrito no (item v.), a operação de junção obriga a especificação de uma condição, o que iguala a sintaxe dos comandos DIALOG das transações T8 e T9. Entretanto, a utilização de arquivos diferentes com massas de dados diferentes permitiu concluir que o tempo de execução de T9 é muito maior. Isto se deve a variação no número de registros de cada arquivo. Um aumento na massa de dados acarretou um aumento de 36 vezes no número de comparações e um aumento quase proporcional (27 vezes) no tempo de execução (vide benchmarks, massas de dados e tempos de resposta nos apêndices C, D, e E, respectivamente).

Ti/Ti	Texec	Toperação	Produtividade	Freq	ProdEsp
T1a	18,563	8,563	1,776	0,152	0,2699
T1b	21,200	3,200	0,312	0,152	0,0474
T2	20,400	2,400	0,417	0,040	0,0167
T3	35,343	57,343	0,058	0,206	0,0119
T4	19,995	1,995	8,501	8,846	8,8238
T5	31,678	13,678	0,073	0,016	8,8812
T6	@8,388	8,388	2,632	0,000	0,0000
"17	365,580	347,580	8,083	8,888	0,0000
T8	2a,727	7,727	8,129	0,063	0,0081
T9	22,235	4,235	0,236	0,818	8,8824
T10	18,451	0,451	2,217	8,888	0,0000
(i .	18,745	8,745	1,342	0,062	0,0832
T12	18,917	8,917	1,091	0,018	0,0196
T13	iEl,611	0,611	1,637	0,026	0,0426
9 . 4	19,454	1,454	8,688	0,118	0,0812
T15	2E?,2@3	18,293	0,098	0,044	0,0043
T16	18,577	0,577	9,733	0,018	0,0312
T17	22,222	4,222	0,237	0,014	0,0033
T18	32,743	94,743	0,068	8,853	0,0010
Total				1,888	8,6470

Obs:
1) Improdutividade = 12s (carga Oracle) + 6s (carga SQL) = 18s
2) Definições nos itens a, d, e, g, ti do item V.3

Figura V.8 - RESULTADOS DE DESEMFENHO DAS TRANSAÇÕES TíPICAS NO ORACLE (em segundos)

A implementação dos benchmarks no ORACLE enfrentou algumas dificuldades por falta de recursos na linguagem. A seguir, serão apresentadas algumas considerações sobre os recursos disponíveis no SGBD.

i. A ativação do ORACLE constrói o ambiente operacional e carrega o SGBD na memória. Este permanece residente até que seja retirado pelo usuário. A execução de aplicações é feita pela ativação da linguagem SQL*PLUS que não permanece em memória ao término da mesma. Assim, elimina-se a necessidade de carregá-lo a cada execução e

não impede-se a execução de outros produtos no computador desde que estes não necessitem ficar residentes em memória.

ii. A SQL é uma linguagem de consulta. A execução de transações possui uma improdutividade relativa a impressão dos registros selecionados em disco, tela ou impressora. Não há como executá-las sem impressão. Logo, consultas que produzem vários registros como resposta, possuem tempo de improdutividade elevado, o que prejudica o tempo de execução.

Os tempos de operação apresentados no quadro relativo ao ORACLE, incorporam a improdutividade relativa a impressão dos registros. Não há como descontá-la, pois seria necessário desenvolver procedimentos que interceptassem as impressões para medí-las.

iii. A carga de tabelas com dados previamente armazenados em arquivos externos, somente, pode ser feita pelo utilitário SQL*LOADER. Não existe comando SQL para realizá-la.

iv. O ORACLE não permite armazenar resultados de consultas em tabelas temporárias ou permanentes. Deve-se, então, armazená-los em arquivos sequenciais em disco, criar tabelas e carregá-las com o SQL*LOADER.

Ti/Tj	Texec [2]	Toperação [2]	Produtividade [2]	Freq [2]	ProdEsp [2]
T5.a	7,400	0,400	2,500	0,152	0,3800
T1.b	8,200	1,200	0,833	0,152	0,1266
T2	—	—	—	0,040	—
T3	8,321	1,321	0,757	0,206	0,1559
r '	—	—	—	0,046	—
T5	7,817	0,817	1,224	0,016	0,0196
T6	—	—	—	0,000	—
r :	—	—	—	0,000	—
T8	—	—	—	0,063	—
'19	—	—	—	0,010	—
T10	—	—	—	0,000	—
T11	—	—	—	0,062	—
T12	—	—	—	0,018	—
T5.3	7,526	0,526	1,901	0,026	0,0494
T5.4	—	—	—	0,118	—
T5.5	8,961	1,961	0,510	0,044	0,0224
T1.6	—	—	—	0,018	—
T17 1	—	—	—	0,014	—
T18	—	—	—	0,015	—
Total				1,000	0,7539

Obs:
1) Improdutividade = 7s
2) Definições nos itens a, d, e, g, h do item V.3

Figura V.9 - RESULTADOS DE DESEMPENHO DA3 TRANSAÇÕES TíPICAS NO PARADOX (em segundos)

A implementação dos benchmarks no PARADOX enfrentou algumas dificuldades por falta de recursos na linguagem. A seguir, serão apresentadas algumas considerações sobre os recursos disponíveis no SGBD.

i. Comandos de manipulação de arquivos não duplicam registros no banco de dados. A inserção de registros cuja chave já existe no arquivo, substitui os dados existentes nos mesmos pelos novos dados. Assim, não há duplicação de chaves, mas tentativas de inclusão de registros já presentes no banco de dados serão aceitas,

o conteúdo original dos mesmos será alterado sem envio de mensagens ao usuário e, consequentemente, serão perdidas as informações previamente armazenadas nestes.

A solução proposta r manuais, é verificar R existência de registros em arquivos antes de inserí-los cru al1erá-los. Deve-se enviar mensagens de confirmação ao uruário de for.ma a evi4ar perdia de dados.

ii. Comandos de manipulação de banco de dados que geram novos arquivos, também, não verificam a existência dos mesmos antes de gerá-los. A substituição do conteúdo é automática nos casos em que o destino já existe. Assim, tentativas de geração de arquivos existentes no banco de dados serão aceitas, o conteúdo original dos mesmos será substituído sem emissão de mensagens ao usuário e, as informações previamente armazenadas serão perdidas.

A solução proposta pelos manuais é a mesma do item (i), isto é, verificar a existência do arquivo antes de gerá-lo. Deve-se enviar mensagens de confirmação ao usuário de forma a evitar perda de informações.

iii. Algumas operações disponíveis no modo interativo através da seleção de opções em menus, não estão implementados diretamente na linguagem PAL. Para executá-las, seria necessário simular o encadeamento das telas nos procedimentos PAL, quando possível.

Consequentemente, a execução de uma operação relacional não seria imediata, envolvendo a construção de procedimentos para simular seu funcionamento.

Entretanto, estes seriam mais lentos, e requisitariam maior programação, sendo compostos de vários comandos. Estes não serão implementados nesta tese por fugirem aos objetivos da mesma que analisa o suporte do produto as transações Ti através da disponibilidade de comandos para realizar as operações relacionais básicas.

iv. A linguagem PAL não dispõe de comandos para implementar as operações relacionais de projeção, interseção, divisão, produto cartesiano e junção (transações T2, T4, T6, T7 e T8). Os comandos básicos existentes permitem, somente, realizar as operações de seleção, união e subtração, mas de forma limitada. Deste modo, verifica-se que a linguagem PAL não é relacional pois não dispõe de recursos que suportem as operações básicas da álgebra relacional.

Assim, seria necessário simulá-la através da construção de procedimentos. Porém, os comandos disponíveis possuem uma área de atuação limitada, dificultando o desenvolvimento dos mesmos. Uma alternativa seria a implementação destes numa linguagem de terceira geração. Contudo, deve-se acrescentar que estes seriam lentos e requisitariam maior programação, fugindo o escopo deste trabalho, conforme descrito no item (iii).

v. Como consequência do item (iv), descrito anteriormente, temos que as transações T9, T10, T11, T12, T14, T16, T17 e T18 não puderam ser implamentadas. A falta de recursos para implementar as sub-transações componentes destas, inviabilizou a codificação das mesmas.

vi. Transações de atualização de campos de arquivo ("UPDATE") devem ser realizadas no modo interativo por menus. A linguagem PAL não dispõe de comandos diretos para realizar tais transações. Uma solução seria simular, quando possível, o encadeamento das telas de menus em procedimentos PAL. Porém, estes não seriam imediatos, seriam lentos e fugiriam aos objetivos desta tese, conforme descrito anteriormente no item (iii).

vii. Os campos numéricos são fixos e podem conter números de até 17 dígitos. Não é permitido definir o tamanho destes campos, sendo alocada a área fixa estabelecida pelo PARADOX. Deste modo, campos pequenos (menos de 17 dígitos) alocam espaço desnecessário enquanto campos longos (mais de 17 dígitos) devem ser definidos como alfanuméricos de forma a poderem ser especificados.

viii. A linguagem PAL limita-se a extender e otimizar programas em aplicações existentes no banco de dados. Não é possível desenvolver completamente uma aplicação em PAL, sem utilização das telas de menu. Isto se deve ao fato de algumas operações disponíveis no modo interativo não estarem implementadas na linguagem PAL.

ix. O comando de união de arquivos (ADD) não gera um arquivo em disco para armazenar os registros resultantes da operação. Os registros de um arquivo são armazenados no final do outro. Assim, o conteúdo original é alterado e perde-se as informações previamente armazenadas.

Uma solução é gerar cópia do arquivo antes de realizar a operação. Assim, conserva-se os dados, permitindo

manipulações futuras. Não é permitida a união de arquivos com tipos de dados diferentes. Os dois arquivos devem ser união-compatíveis.

Ti/Tj	Texec	Toperação	Produtividade	Freq	ProdEsp
T1a	6,180	0,180	5,556	0,152	0,8445
T1b	7,357	1,357	0,737	0,152	0,1120
T2	7,592	1,592	0,628	0,040	0,0251
T3	10,563	4,563	0,219	0,206	0,0451
T4	9,387	3,387	0,295	0,046	0,0136
T5	8,137	2,137	0,468	0,016	0,0075
T6	–	–	–	0,000	–
T7	39,527	33,527	0,030	0,000	0,0000
T8	8,138	2,138	0,468	0,063	0,0295
T9	11,674	5,674	0,176	0,010	0,0018
T10	7,158	1,158	0,864	0,000	0,0000
T11	6,957	0,957	1,045	0,062	0,0648
T12	6,563	0,563	1,776	0,018	0,0319
T13	6,956	0,956	1,046	0,026	0,0272
T14	11,676	5,676	0,176	0,118	0,0208
T15	14,237	8,237	0,121	0,044	0,0053
T16	13,273	7,273	0,137	0,018	0,0025
T17	11,853	5,853	0,171	0,014	0,0024
T18	11,861	5,861	0,171	0,015	0,0026
Total				1,000	1,2366

Obs:
1) Improdutividadc = 6ç
2) Definições nos itens a, d, e, g, h do item V.3

Figura V.10 – RESULTADOS DE DESEMPENHO DAS TRANSAÇÕES TÍPICAS NO RBASE SYSTEM V (em segundos)

A implementação dos benchmarks no RBASE SYSTEM V enfrentou algumas dificuldades por falta de recursos na linguagem. A seguir, serão apresentadas algumas considerações sobre os recursos disponíveis no SGBD.

i. O comando de união de arquivos (UNION) não gera um arquivo em disco para armazenar os registros resultantes

da operação. Os registros de um são armazenados no final do outro, sem eliminação dos repetidos. Deste modo, o conteúdo original é alterado e perde-se as informações previamente armazenadas.

Uma solução é gerar cópia do arquivo antes de realizar a operação. Assim, conserva-se os dados, permitindo manipulações futuras.

Em termos de E/S, a quantidade realizada pelo comando é praticamente a mesma daquela dos outros comandos que geram outro arquivo em disco. A criação/inserção de registros requer espaço em memória secundária para armazená-los, condicionando seu sucesso a existência de espaço suficiente para seu armazenamento.

é permitida a união de arquivos com tipos de dados diferentes. Para tanto, apenas os campos existentes em ambos são copiados, não precisando que estes estejam na mesma ordem, mas que tenham o mesmo nome. Logo, os campos do arquivo fonte não comuns ficam sem informação e o arquivo resultante fica incompleto.

Entretanto, a obrigatoriedade dos campos comuns terem o mesmo nome prejudica a manipulação de arquivos com mesmo tipo de dados, mas nomes de campos diferentes. é necessário, então, gerar os arquivos com nomes de campos iguais de forma a permitir operações futuras.

ii. Os comandos de união (UNION), interseção (INTERSECT) e subtração (SUBTRACT) obrigam que os campos comuns nos arquivos a serem manipulados tenham o mesmo nome.

Arquivos com tipo de dados idêntico mas nomes de campos diferentes não podem ser comparados. Portanto, deve-se gerar as tabelas de forma que os campos a serem comparados tenham mesma identificação.

iii. Os comandos de manipulação de arquivos obrigam a criação de um outro arquivo em disco para armazenar os registros resultantes da operação. Deste modo, as operações relacionais tornam-se lenta e pouco eficiente, e não possuem os objetivos reais fixados na álgebra relacional. Uma das principais causas é a quantidade de E/S envolvida na criação e inserção de registros no novo arquivo. Adicionalmente, a execução requer espaço em disco, condicionando seu sucesso a existência de quantidade de memória secundária suficiente para o armazenamento do arquivo.

Deveria ser permitido a geração de arquivos temporários que seriam deletados ao fim de cada sessão RBASE, caso não fossem transformados em permanentes.

iv. A operação relacional de junção (JOIN) obriga a comparação entre um campo de cada arquivo que tenham nomes diferentes entre si. Adicionalmente, não é permitido a especificação de várias condições. Assim, a operação fica restrita, não sendo possível representar todas as situações reais de uma instalação. Os manuais dizem que estas restrições visam evitar confusões, pois haveria vários tipos possíveis de junção.

Deste modo, a realização de uma operação que deva satisfazer várias condições, deve ser fragmentada em

vários comandos de forma a obedecer as restrições impostas pela linguagem.

v. A linguagem não dispõe de recursos que permitam realizar o produto cartesiano de tabelas. A solução encontrada foi utilizar o comando JOIN com a especificação de uma condição que produza valor verdadeiro como resultado, em todas as situações. Isto se deve ao fato do comando obrigar a especificação de uma condição envolvendo um campo de cada arquivo, que tenham nomes diferentes e, tipo e comprimento iguais.

Esta alternativa torna mais lenta a operação, pois cada novo registro lido, obriga o teste da condição. Assim, consegue-se realizar a operação relacional, contudo, esta possui desempenho inferior a de outros SGBD's

vi. Comandos de manipulação de arquivos (APPEND, PROJECT e CHANGE) podem duplicar registros no banco de dados. Há um comando (DELETE DUPLICATE) para eliminar as linhas duplicadas em tabelas. Cabe ao usuário executá-lo.

vii. Alguns comandos (CHANGE, DELETE e JOIN) obrigam a especificação da cláusula de condição (WHERE). Nos casos em que a operação deve ser realizada em todos os registros, especifica-se uma condição que seja verdadeira para todas as linhas da tabela. Porém, o uso de condições abrangentes aumenta o tempo de execução da operação e torna o processamento mais lento.

viii. Não é [illegible] especificar o comprimento dos campos numéricos ("integer", "real" e "double"). Os números são

armazenados no formato binário, respeitando a precisão e comprimento máximo de cada tipo de dados no RBASE. Não conseguiu-se obter informações relativas ao espaço de armazenamento alocado a cada campo durante a sua definição. Não há informação na bibliografia utilizada nesta tese, que permita saber se os campos possuem comprimento fixo e igual ao máximo no RBASE ou, possuem comprimento variável e igual ao do dado armazenado correntemente no banco de dados, respeitando o limite máximo do SGBD.

Ti/Tj	Texec [2]	Toperação [2]	Produtividade [2]	Freq [2]	ProdEsp [2]
T1a	7,613	0,613	1,631	0,152	0,2479
T1b	7,957	0,957	1,045	0,152	0,1588
T2	–	–	–	0,040	–
T3	8,378	1,378	0,726	0,206	0,1495
T4	8,378	1,378	0,726	0,046	0,0333
T5	8,378	1,378	0,726	0,016	0,0116
T6	–	–	–	0,000	–
T7	–	–	–	0,000	–
T8	9,006	2,006	0,499	0,063	0,0314
T9	10,201	3,201	0,312	0,010	0,0031
T10	8,433	1,433	0,698	0,000	0,0000
T11	8,001	1,001	0,999	0,062	0,0619
T12	–	–	–	0,018	–
T13	8,285	1,285	0,778	0,026	0,0202
T14	8,823	1,823	0,549	0,118	0,0648
T15	8,433	1,433	0,698	0,044	0,0307
T16	9,375	2,375	0,421	0,018	0,0076
T17	–	–	–	0,014	–
T18	–	–	–	0,015	–
Total				1,000	0,8208

Obs:
1) Improdutividade = 7s
2) Definições nos itens a, d, e, g, h do item V.3

Figura V.11 - RESULTADOS DE DESEMPENHO DAS TRANSAÇÕES TÍPICAS NO ZIM (em segundos)

A implementação dos benchmarks no ZIM enfrentou algumas dificuldades por falta de recursos na linguagem. A seguir, serão apresentadas algumas considerações sobre os recursos disponíveis no SGBD.

i. A operação relacional de junção quando realizada no SGBD ZIM não elimina a segunda ocorrência de atributos comuns nas tuplas resultantes.

ii. A linguagem ADL não permite implementar a operação relacional de divisão, apesar de possuir estrutura próxima do padrão relacional. Deste modo, seria necessário desenvolver um procedimento em linguagem de terceira geração para simular esta operação.

Entretanto, estes procedimentos seriam lentos, e requisitariam maior programação, sendo compostos de vários outros comandos. Este não é o objetivo deste trabalho que visa analisar o desempenho de SGBD's através da execução de benchmarks construídos incorporando as transações relacionais mais frequentes de uma instalação. Assim, deve-se analisar o suporte do produto as transações Ti, a disponibilidade de comandos para realizar as operações relacionais básicas e o desempenho destes em relação ao de outros produtos.

iii. O conceito de projeção disponível na linguagem, somente, pode ser usado juntamente com operações que envolvam várias entidades (ex: junção). O objetivo é gerar resultados com campos da apenas uma das envolvidas na operação, isto é, projetar o resultado sobre uma das entidades. Logo, não consegue-se realizar a operação

simples de projeção, impedindo a seleção de um sub-conjunto de atributos de uma entidade.

iv. Os comandos de união (UNION), interseção (INTERSECT) e subtração (MINUS) obrigam a utilização de uma única entidade durante a operação e de comandos de seleção para buscar os elementos desta entidade a serem comparados entre si. O objetivo destes comandos é substituir as operações que fazem uso de operadores relacionais (OR, AND e NOT), otimizando a execução destas através da definição de índices na entidade e do uso destes na geração do resultado.

Logo, estes comandos não atendem aos objetivos das operações relacionais de mesmo nome. Entidades diferentes com tipos de dados iguais não podem ser manipuladas, independentemente do nome de seus atributos. Perde-se as vantagens destas operações, não conseguindo realizá-las sobre várias entidades e sem associações com comandos de seleção (FIND's).

Adicionalmente, não consegue-se realizar transações que usam estas operações (união, subtração e interseção) juntamente com outras envolvendo várias entidades (ex: junção, projeção). Assim, transações mais complexas tornam-se inviáveis de serem implementadas.

v. A operação relacional de junção fica restrita a existência de pelo menos um relacionamento entre as entidades que a compõem. Sua implementação é feita pelo comando de seleção (FIND) que obriga a declaração do nome do relacionamento através do qual os registros

estão associados. Seu resultado contém, apenas, os registros que atendem a condição de relacionamento definida quando da criação do mesmo. Deste modo, não é possível concatenar tuplas de qualquer duas entidades de forma que as tuplas resultantes satisfaçam a condição especificada no comando.

A execução desta operação não elimina a segunda ocorrência dos atributos comuns das duas entidades (atributos com mesmo domínio básico e mesmo nome), tornando-os repetitivos nas tuplas resultantes. Estas tuplas passam a conter todos os atributos das duas entidades envolvidas na operação, independentemente destes possuírem mesmo nome e tipo (domínio), pois a operação citada não é uma junção natural.

vi. As restrições descritas no (item v.) impedem a implementação da operação relacional de produto cartesiano. Esta precisa concatenar todas as ocorrências de duas entidades entre si e não apenas aquelas que satisfazem a condição de relacionamento. Deste modo, não pode ser simulada pelo comando FIND.

V.5 DESEMPENHO DOS SGBD's EM CADA BENCHMARK DE BAIXO NíVEL

A execução dos benchmarks em cada SGBD gerou resultados de desempenho. A partir dos tempos de operação em segundos de cada transação Ti em cada SGBD gerou-se quadros que permitiram analisar o desempenho de cada produto em Ti.

Ti/SGBD	COPPEREL	DBASE	DIALOG	ORACLE	PARADOX	RBASE	ZIM
T1a	0,486	0,143	0,235	0,563	0,400	0,180	0,613
T1b	0,807	0,603	0,468	3,200	1,200	1,357	0,957
T1'2	43,878	0,532	0,487	2,488	-	1,592	-
T3	2,317	20,920	15,171	17,343	1,321	4,543	1,378
T4	0,745	4,026	5,672	1,995	-	3,387	1,378
T5	2,325	-	-	13,678	0,817	2.137	1,378
T6	2,285	-	-	8.380	-	-	-
T7	1,792	7,286	3,881	347,580	-	33,587	-
T1'8	1,593	1,593	1,600	7,727	-	2,138	2,006
T9	6,973	75,844	53,818	4,538	-	5,674	3,201
T10	0,801	0,368	0,266	0,451	-	1,158	1,433
T11	0,801	1,111	0,811	8,745	-	0,957	1,001
T12	1,371	0,610	8,688	0,917	-	0,583	-
T13	0,403	0,979	0,641	0,611	0,526	0,956	1,285
T14	6,981	75,830	53,018	1,454	-	5,676	1,823
T15	2,527	3,880	1,408	10,.13	5,961	8,237	1,433
T16	7,912			0,577	-	7,273	2,375
T17	3,417	-	-	4,222	-	5,853	-
T18	4,678	3,255.	2,999	14,743	-	5,861	-
Total	49,792	329,707	212,266	433,834	6,225	91,089	20,261

Figura V.12 - TEMPO DE OPERAÇÃO DE CADA Ti EM CADA SGBD

Os resultados obtidos permitiram concluir que:

i. A transação T1a é a mais rápida na maioria dos SGBD's.

ii. O COPPEREL e o ORACLE são os únicos com recursos para executar todas as operações relacionais. Não construiu-se procedimentos em linguagem de terceira geração. Os comandos disponíveis para em propósitos relacionais e, assim, puderam implementar todas as transações Ti.

iii. O PARADOX possui linguagem com menor poder relacional. Logo, pode-se implementar, somente, seis (6) transações.

iv. O COPPEREL possui desempenho semelhante em todas as transações. Por exemplo, na base de dados em teste, o tempo de operação foi inferior a 8 (oito) segundos.

v. A operação relacional de divisão (transação T6) pode ser implementada, somente, no COPPEREL e ORACLE.

vi. O ORACLE teve seu tempo de operação prejudicado em virtude da incorporação do tempo de improdutividade relativo a impressão dos resultados das consultas na tela. Não foram desenvolvidos procedimentos para interceptar as impressões e medir este tempo.

Observadas as frequências das transações Ti (figuras V.1 e V.2), montou-se um quadro do toper médio esperado em segundos (esperança) de cada transação típica em cada SGBD.

Ti/SGBD	COPPEREL	DBASE	DIALOG	ORACLE	PARADOX	RBASE	ZIM
T1	0,074	0,022	0,036	0,086	0,061	0,027	0,093
T2	0,123	0,092	0,071	0,486	0,182	0,206	0,145
T3	0,035	0,021	0,019	0,096	–	0,064	–
T4	0,477	4,309	3,125	3,573	0,272	0,939	0,284
T5	0,034	0,185	0,261	0,092	–	0,156	0,063
T6	0,037	–	–	0,219	0,013	0,034	0,022
T7	0,000	–	–	0,000	–	–	–
T8	0,000	0,000	0,000	0,000	–	0,000	–
T9	0,100	0,100	0,101	0,487	–	0,135	0,126
T10	0,069	0,758	0,530	0,042	–	0,057	0,032
T11	0,000	0,000	0,000	0,000	–	0,000	0,000
T12	0,049	0,069	0,050	0,046	–	0,059	0,062
T13	0,025	0,011	0,011	0,017	–	0,010	–
T14	0,011	0,025	0,017	0,016	0,014	0,025	0,033
T15	0,824	8,948	6,256	0,172	–	0,669	0,215
T16	0,111	0,132	0,062	0,449	0,086	0,362	0,063
T17	0,142	2,405	1,296	0,010	–	0,131	0,043
T18	0,048	–	–	0,059	–	0,082	–
	0,070	0,049	0,045	0,221	–	0,088	–
Toper	2,229	17,126	11,880	6,071	0,628	3,044	1,181

Figura V.13 – TEMPO DE OPERAÇÃO MÉDIO ESPERADO DE CADA TRANSAÇÃO TÍPICA EM CADA SGBD

Os resultados mostrados na figura V.12 permitiram construir um gráfico do desempenho de cada transação Ti em cada SGBD, que será apresentado na figura V.14. No eixo das abscissas, serão representadas as transações Ti, que são 19, e no eixo das ordenadas será marcado o tempo de operação em segundos de cada Ti no SGBD em questão.

V.6 EXECUÇÃO DO BENCHMARK DE ALTO NÍVEL E ANÁLISE DE DESEMPENHO DOS SGBD's

A partir da aplicação genérica AEI e dos benchmarks de baixo nível, foi definido e implementado um benchmark de alto nível T que englobou as transações Ti em um único programa. Considerou-se a distribuição de frequência das aplicações da Eletrobrás e, os requisitos e objetivos da AEI. Assim, combinou-se as transações Ti de baixo nível obedecendo os mesmos critérios de busca, usou-se os mesmos recursos de Ti, analisou-se os mesmos parâmetros dinâmicos (confirmando as considerações apresentadas no item V.3), e respeitou-se as deficiências dos SGBD's que impediram a implementação de algumas transações em T.

O benchmark T foi executado várias vezes de forma a gerar parâmetros dinâmicos corretos e seguros. Assim, analisou-se os resultados de desempenho de cada SGBD.

A partir destes resultados, construiu-se um quadro da distribuição do tempo de operação pelo número de execuções de "T" em cada SGBD, que é apresentado no apêndice E (figura E.8). Fixou-se como tempo de operação, o intervalo de tempo

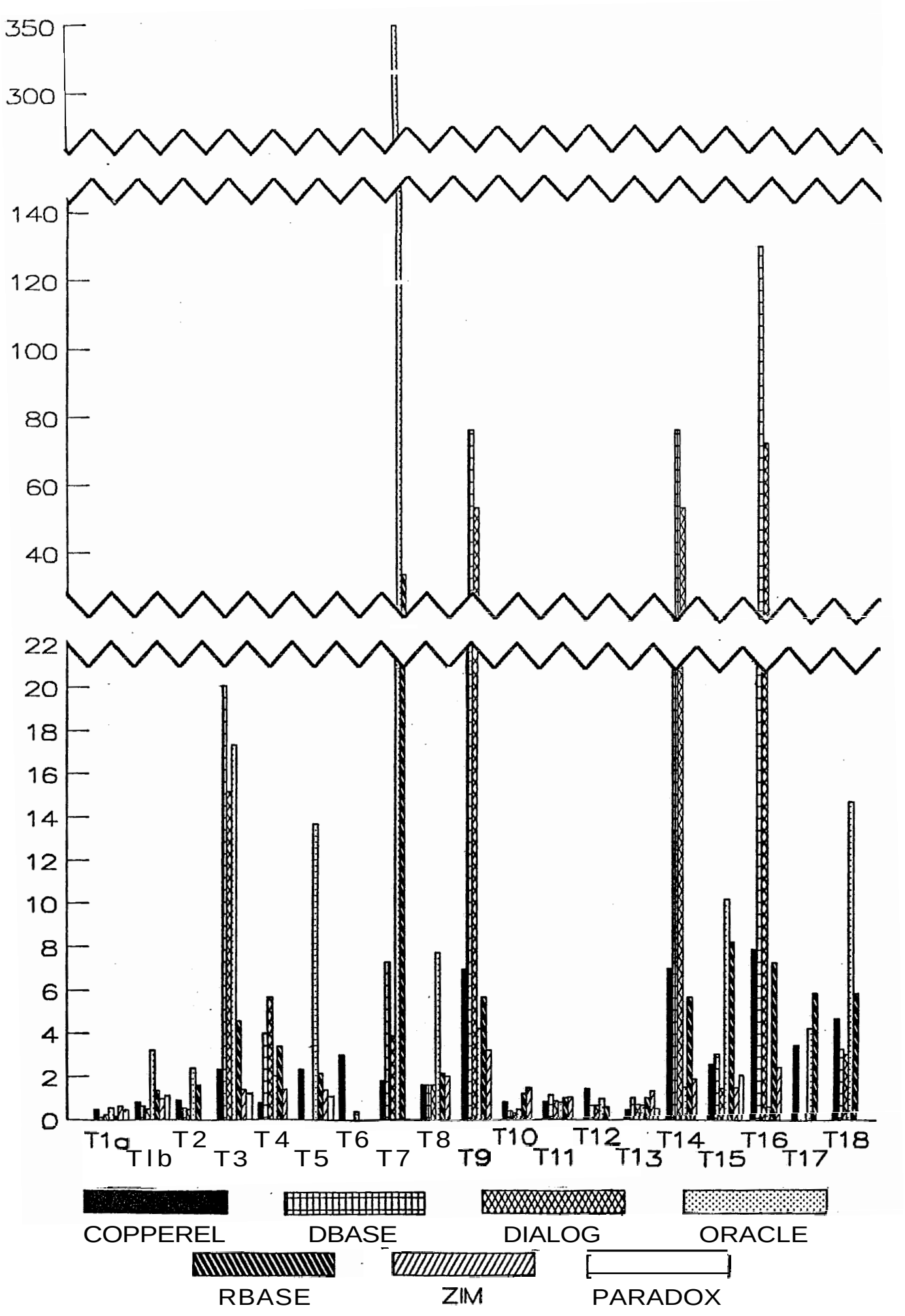

Figura V.14 - TEMPO DE OPERAÇÃO DE CADA Ti EM CADA SGBD

necessário para processar T. Logo, este tempo coincidiu com o tempo de execução, descontado o tempo de improdutividade.

Em seguida, construiu-se um quadro do tempo de operação médio de T em cada SGBD, que permitiu analisar o desempenho de cada produto em T. A seguir, será apresentado este quadro com os valores de Toper em segundos.

Deve-se considerar que algumas transações não puderam ser implementadas em T devido a deficiências de alguns produtos. Consequentemente, os tempos de operação obtidos ficaram distorcidos e sem significado, por não retratarem a execução das mesmas transações em todos os SGBD's.

T1/ SGBD	COPPEREL	I ASE	DIALOG	ORACLE	PARADOX	RBASE	ZRM
T	1113,102	1223,735	861,773	393,983	57,549	206,836	81,878

Figura V.15 - TEMPO DE OPERAÇÃO DE T EM CADA SGBD

Os resultados obtidos permitiram construir um gráfico do desempenho de T em cada SGBD. A seguir, este será apresentado (figura V.16). No eixo das abscissas, serao representados os SGBD's, que são 7(sete), e no eixo das ordenadas será marcado o tempo de operação em segundos de cada SGBD com relação a T.

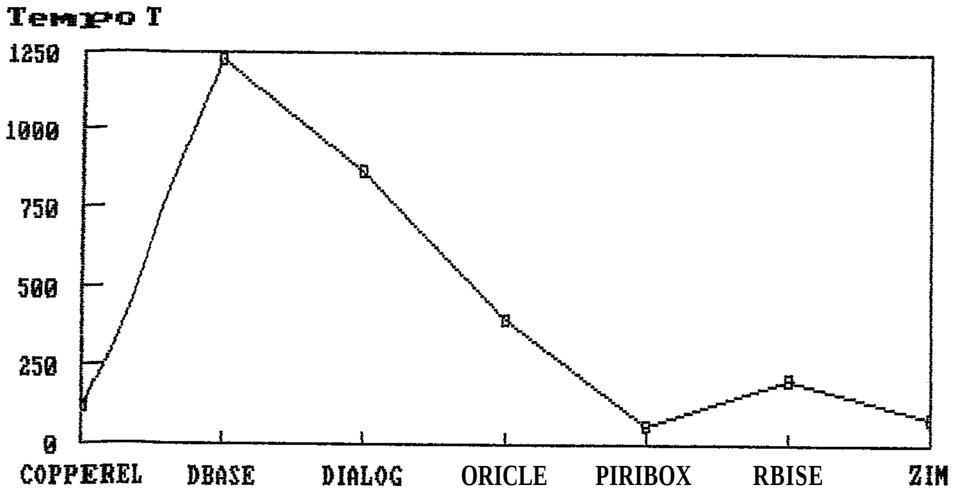

Figura V.16 - TEMPO DE OPERAÇÃO DE T EM CADA SGBD

Em decorrência das dificuldades encontradas durante a implementação de T em todos os SGBD's, decidiu-se analisar e validar os resultados de desempenho obtidos de T, apenas, para o COPPEREL E ORACLE. Estes produtos são os únicos capazes de implementar e executar todas as transações Ti. A seguir, será apresentado um quadro do tempo de operação médio de T em segundos nestes dois SGBD's.

Ti/SGBD	COPPEREL	ORACLE
T	113,[illegible]	393,983

Figura V.17 - TEMPO DE OPERAÇÃO DE T EM CADA SGBD

Os resultados mostrados acima, são significativos e comparáveis. Todas as transações Ti foram executadas sem restrições e permitiram analisar o desempenho real do COPPEREL e ORACLE.

A construção e implementação do benchmark de alto nível T ficou limitada pelas restrições impostas por alguns

produtos. A falta de recursos de linguagem impediram a incorporação de algumas transações Ti em T. A seguir, será apresentado um quadro do número de transações implementadas e não implementadas em cada SGBD.

SGBD	NO. TRANSAÇÕES IMPLEMENTADAS	No. TRANSAÇÕES NÃO IMPLEMENTADAS	TRANSAÇÕES NÃO IMPLEMENTADAS
COPPEREI	59	08	.....
DBASE	16	03	T5, T6, T5.7
DIALOG	16	03	T5, T6, T17
ORACLE	19	00	.....
PARADOX	06	13	T2, T4, T6, T7 T8, T9, T10, T11, T12, T14, T16, T17, T18
RBASE	18	1	T6
ZIM	13	6	T2, T6, T7, T12 T17, T18

Figura V.18 - IMPLEMENTAÇÃO DAS TRANSAÇÕES POR SGBD

Pelo fato do PARADOX permitir a implementação de apenas 6 (seis) transações típicas, a execução do benchmark de alto nível neste SGBD apresentou o menor tempo de operação (Toper). Este fato comprova que, em alguns casos, um benchmark de alto nível pode gerar resultados de desempenho distorcidos. Neste caso, o produto com menor capacidade relacional apresentou-se no gráfico como o mais eficiente.

Logo, estes resultados não são válidos para analisar o desempenho dos SGBD's. Conforme descrito anteriormente, a implementação de um benchmark em produtos com escopos distintos e linguagens restritas gera resultados distorcidos. Este não representa com exatidão a aplicação AEI e a distribuição de frequência das transações na

Eletrobrás, tem seu escopo limitado em alguns produtos com a impossibilidade de representar algumas Ti, não permite a comparação dos resultados entre si por estes não serem representativos da realidade dos produtos, e gera tempos de operação reduzidos nos SGBD's limitados.

Assim, conclui-se que a análise de desempenho destes SGBD's no ambiente em questão deve considerar, principalmente, os resultados gerados pelos benchmarks de baixo nível e ignorar aqueles gerados por T. O alto grau de diferenciação entre os produtos quanto a capacidade e funcionalidade prejudicou a construção de T. Este fato permitiu observar as dificuldades citadas no capítulo II em ambientes reais.

Decidiu-se, então, construir um benchmark de alto nível Tmin contendo o subconjunto máximo de transações Ti implementáveis em todos os SGBD's. Este é formado por 5 das 6 transações Ti implementadas no PARADOX. Segundo definido anteriormente, não seriam usados recursos externos, mas somente recursos e linguagens disponíveis nos produtos, para implementar os benchmarks. Assim, o benchmark Tmin incorporou apenas as transações T1a, T1b, T3, T13 e T15.

O benchmark Tmin ficou limitado e não retratou com exatidão a distribuição de frequência das aplicações na Eletrobrás e a aplicação AEI. Porém, gerou resultados comparáveis entre si e significativos. A seguir, será apresentado um quadro do tempo de operação médio de Tmin em cada produto. Os valores de Toper são apresentados em segundos.

T'i/	COPPEREL	DBASE	DIALOG	ORACLE	PhRADOX	RBASE	ZIM SGBD
T	53,243	326,576	216,239	321,531	53,427	112,999	53,666

Figura V.19 - TEMPO DE OPERAÇÃO DE Tmin EM CADA SGBD

Estes resultados mostram que o toper obtido por Tmin é bem próximo ao Toper obtido por T no PARADOX, pois Tmin implementa 5 das 6 transações de T no PARADOX. Assim, comprova-se que os números obtidos em T são distorcidos e não significativos para analisar o desempenho dos SGBD's.

A seguir, será apresentado um gráfico do desempenho de Tmin nos SGBD's. No eixo das abscissas serão representados os SGBD's e no eixo das ordenadas serão marcados os tempos de operação médio em segundos destes em Tmin.

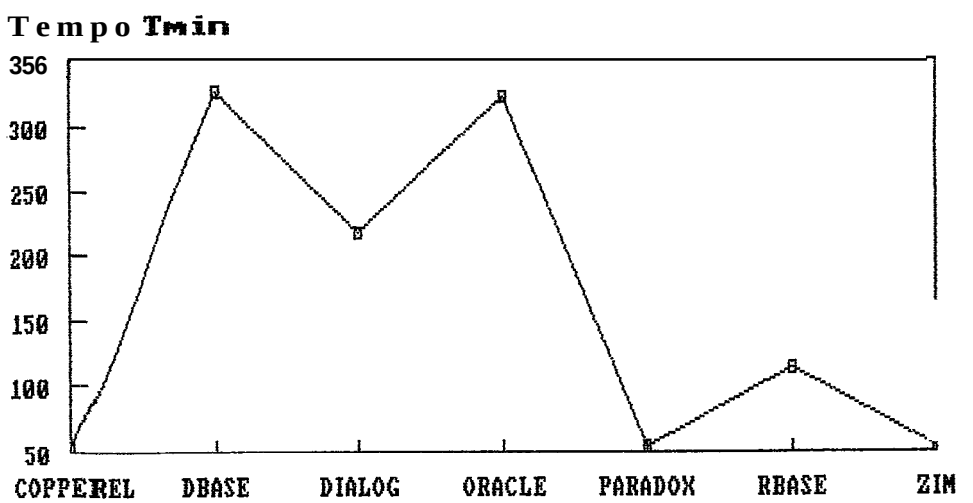

Figura V.20 - TEMPO DE OPERAÇÃO DE Tmin EM CADA SGBD

CAPÍTULO VI

OTIMIZAÇÃO DO COPPEREL

Neste capítulo, serão descritas deficiências identificadas no COPPEREL. Para tanto, serão apresentadas propostas de melhorias para o mesmo visando otimizar o seu desempenho.

VI.1 DEFICIÊNCIAS E PROPOSTAS DE MELHORIAS

O COPPEREL, como os demais produtos, possui aspectos positivos e negativos em suas características. No apêndice A, estas são apresentadas, permitindo melhor conhecer o SGBD. Assim, neste capítulo, serão descritas somente as deficiências do mesmo, pois as características positivas podem ser obtidas no apêndice.

Os resultados de desempenho obtidos a partir da análise dos parâmetros estáticos e dinâmicos para o COPPEREL, mostram que o desempenho deste deve ser otimizado através da incorporação de novas características e ferramentas. Foram identificadas deficiências que precisam ser eliminadas através de extensões e modificações no mesmo, de forma a melhorar seu desempenho e funcionalidade.

A seguir, serão listadas deficiências encontradas na funcionalidade do COPPEREL e propostas melhorias para o mesmo. A fim de facilitar a descrição das deficiências, a sequência em que serão relatadas segue a ordem de apresentação dos parâmetros estáticos no capítulo III. Será

feita uma descrição sucinta do problema e apresentada uma proposta de melhoria para o mesmo.

a. Estrutura física e organização do espaço de armazenamento

Os arquivos não permitem a definição de regras de validação associadas aos campos para a conferência dos valores fornecidos pelo usuário quanto a comprimento, formato e conteúdo. Isto permite o cadastramento de informações inválidas e inconsistentes, pois a entrada de dados em arquivos não é criticada. Assim, deve ser possível a definição de regras de validação associadas aos campos no arquivo físico. Estas devem permitir a inclusão de comandos Loperel na sua especificação, ficar armazenadas no dicionário de dados e estar associadas aos comandos que tratam da entrada de dados.

b. Tipos de dados

O tipo de dados "Data" foi definido na especificação do COPPEREL, entretanto, ainda não foi implementado. Deve-se colocá-lo em funcionamento, pois a maioria das aplicações necessita representar dados neste formato.

c. Dicionário de dados

O dicionário de dados deve ser extendido para armazenar o máximo de informações possíveis sobre os arquivos, estruturas, dados, aplicações e usuários. Este deve garantir a integridade e consistência dos dados gerados ao longo das fases do desenvolvimento de aplicações.

Modificações devem ser realizadas em um único local e propagadas para todos os objetos que as utilizam.

d. Linguagem de definição e manipulação de dados

Foram identificadas várias deficiências na linguagem LOPEREL. A seguir, serão feitas algumas considerações.

i. O comando MODIFICAR obriga a especificação de uma cláusula de condição. Quando a operação tem que ser realizada em todos os registros, especifica-se uma condição que seja verdadeira para todos, isto é, não existe o "MODIFICAR TODOS OS REGISTROS". Porém, o uso de condições genéricas torna o processamento mais lento. Há um comando iterativo que permite percorrer todo o arquivo modificando um a um os registros, mas a condição deve ser especificada.

ii. Não existe nenhum mecanismo embutido na linguagem LOPEREL para tentar corrigir erros cometidos pelo usuário na escrita de palavras reservadas ou identificadores. A construção de um dicionário de palavras reservadas ajudaria a detectar erros de sintaxe em comandos, operandos e operadores.

iii. A linguagem não dispõe de um "help" contextual com janelas para ajudar a depurar erros e esclarecer dúvidas sobre a sintaxe dos comandos. Porém, um comando de socorro mostra na tela partes do manual de usuário. A construção de janelas facilitaria o uso do COPPEREL e a codificação de aplicações.

iv. A linguagem não permite consultas aninhadas, sendo necessário desmembrar algumas consultas em vários comandos. Porém, ajuda a usuários, acostumados a programar em linguagens procedimentais, a formular consultas. Modificações na estrutura da linguagem LOPEREL e de seus comandos permitiria aninhamento de comandos e, consequentemente, de consultas.

v. A não existência de telas formatadas de entrada de dados, torna a adição de registros via teclado não flexível, tendo que obedecer o formato e ordem dos campos pré-definidos na criação do arquivo. Assim, registros devem ter seus campos preenchidos na sequência correta e formato preciso para evitar erros de entrada de dados. O desenvolvimento de uma interface com usuário amigável e a incorporação de comandos LOPEREL que permitam a geração de telas de entrada de dados associadas aos arquivos, elimina a obrigatoriedade acima, podendo ser preenchidos somente os campos obrigatórios.

vi. A operação relacional de junção é uma equi-junção, não eliminando a segunda ocorrência de atributos comuns (mesmo domínio básico) das duas entidades. As tuplas passam a conter todos os atributos das duas entidades envolvidas na operação, respeitando o limite fixado pelo COPPEREL para o tamanho de registros de arquivos. Implementando-se, também, a junção natural, eliminaria-se a segunda ocorrência de atributos comuns na tabela resultante, reduzindo o espaço de memória necessário ao armazenamento da

tabela e eliminando a redundância gerada com a criação de duas colunas idênticas.

vii. A execução do COPPEREL via arquivo de comandos externo mostra na tela ou em arquivo o trace de execução dos comandos. Não há como desligar a comunicação com o usuário. Logo, aumenta o tempo de improdutividade provocado pela emissão do trace. Deveria ser permitido ao usuário desligar o envio de mensagens e, assim, sua comunicação com o SGBD.

viii. Algumas operações obrigam a especificação de um arquivo de saída para armazenar os registros resultantes. Em algumas (produto cartesiano e junção), a especificação de um temporário restringe o escopo da operação, podendo esta ser executada, somente, sobre entidades cujos nomes dos atributos sejam distintos. Arquivos que possuem atributos com nomes em comum impedem sua execução. O COPPEREL deveria tratar igualmente os temporários e permanentes a nível de execução de uma transação.

ix. A operação de seleção com intervalo de valores sofre restrições quanto c número de critérios de comparação e a combinação de operadores. Utiliza-se algoritmos de classificação na execução de seleções que envolvim critérios de comparação diferesites da igualdade, para os quais os índices não são dficientes. Deve-se otimizar esta operação de forma a melhorar seu desempenho, expandir sua capacidade e elimisiar suas restrições.

e. Interface com usuário

Deve-se melhorar a interface com o usuário através da construção de menus que permitam a seleção das tarefas a serem realizadas através da escolha de opções. Assim, elimina-se a necessidade de conhecer a sintaxe dos comandos para programar uma aplicação. Se o usuário tiver experiência no COPPEREL e quizer otimizar seu uso, poderá continuar a realizar a entrada de dados e comandos em massa via leitura de arquivos externos.

Telas formatadas de entrada de dados auxiliariam na adição de registros. Não preocuparia-se com o formato dos campos pré-definido na criação do arquivo e não necessitaria-se preencher todos os campos na sequência correta. Uma tela mostraria o lay-out do arquivo e, somente, os campos obrigatórios seriam preenchidos.

Adicionalmente, seria possível gerar telas formatadas de saída que apresentariam dados resultantes de operações em forma de telas otimizadas. A geração de relatórios seria facilitada pela utilização das telas de saída como lay-out dos mesmos.

f. Integração com outros SGBD's

O uso do COPPEREL em instalações que possuem outros produtos de banco de dados, necessita de recursos para migrar aplicações e dados entre estes produtos e o COPPEREL. Logo, deve-se construir, um utilitário que permita importar/exportar dados. Este deve fixar

formatos e produtos que podem ser integrados ao banco de dados da COPPE e regras de conversão para os mesmos.

g. Qualidada da documentação

A documentação do COPPEREL precisa ser melhorada como um todo, principalmente, com relação aos manuais de operação e usuário. Várias extensões e modificações foram realizadas no SGBD e não foram incorporadas aos manuais. Dentre elas, podemos citar: interface para linguagem hospedeira, objetos complexos, campos longos e subsistema de reconstrução. Assim, usuários precisam pesquisar trabalhos e teses realizados na COPPE/SISTEMAS/UFRJ para obter informações e, melhor fazer uso da funcionalidade e características do produto.

h. Segurança e proteção dos dados

Não foi detectada nenhuma deficiência neste parâmetro estático.

i. Deteção de falhas e reconstrução do banco de dados

Os arquivos AI e BI servem para armazenar transações realizadas com sucesso e a imagem dos dados antes de cada transação de atualização, respectivamente, numa sessão. Deste modo, consegue-se recuperar o banco de dados em falhas de transação e de meio.

Porém, estes arquivos AI e BI deveriam ser atualizados ao término de cada sessão com sucesso, para guardar, somente, as informações necessárias para recuperar a sessão corrente. Atualmente, após a abertura/fechamento

de várias sessões é apresentada uma tela de advertência informando o estouro do log BI. Independente da opção escolhida, esta se repete, frequentemente. Deve-se aumentar o tamanho do log BI e avaliar formas de atualizá-lo para mantê-lo sempre com os dados necessários a recuperação da sessão em uso, de forma a evitar o estouro de espaço do log.

Hoje, o objetivo é manter as transações de atualização (realizadas nas sessões abertas após a geração de "back-up" do banco de dados) nos arquivos AI e BI, para poder recuperar o banco de dados em caso de falhas de meio físico. Porém, a cada fim de sessão, estes arquivos deveriam ser atualizados para evitar estouros de log. O usuário deveria, também, tirar cópia dos dados a cada início de sessão e zerar os arquivos de log para eliminar estes problemas.

j. Concorrência

Há uma proposta de extensão multiusuária para o SGBD que ainda não foi implementada. Esta está totalmente especificada em [Pale85] e, deveria ser implementada e colocada em funcionamento em breve, de forma a suportar acessos concorrentes e compartilhados ao COPPEREL e suas bases de dados. Um protótipo deveria ser gerado e colocado em teste em ambientes variados para verificar seu desempenho e funcionalidade.

k. Ambiente operacional

Não foi detectada nenhuma deficiência neste parâmetro estático.

l. Portabilidade e flexibilidade

Não foi detectada nenhuma deficiência neste parâmetro estático. Deve-se acrescentar que o COPPEREL armazena os dados de uma base de dados em um único arquivo físico facilitando a portabilidade do banco de dados para várias máquinas.

m. Gerador de aplicações

O COPPEREL deve permitir a usuários não pertencentes a área de informática e aqueles sem conhecimento da linguagem LOPEREL, construir aplicações de forma rápida e segura. Logo, deve haver um gerador de aplicações para desenvolver e alterar aplicações sem programação, com custo e tempo de desenvolvimento reduzidos. Geradores específicos devem permitir construir relatórios, telas, bases de dados e consultas.

Para viabilizar a construção deste gerador, deve-se, primeiramente, desenvolver uma interface interativa por menus que permita selecionar tarefas em menus, sem conhecimento da sintaxe dos comandos.

n. Arquitetura

Não foi detectada nenhuma deficiência neste parâmetro estático.

o. Utilitários

Conforme descrito no item f, deve-se desenvolver e implementar um utilitário para importar/exportar dados de outros SGBD's. Este deve fixar formatos e produtos que podem usá-lo e regras de conversão para os dados.

Em futuros trabalhos, poderiam ser implementadas as alterações propostas para o COPPEREL, melhorando seu uso.

Quanto aos parâmetros dinâmicos, estes mostraram que o COPPEREL gerou resultados de desempenho muito bom. Estes resultados, na maioria dos casos, são melhores que aqueles gerados por outros SGBD's, em termos de tempo de operação.

Deve-se acrescentar que o COPPEREL e ORACLE foram os únicos produtos com capacidade para implementar todos os benchmarks de baixo nível e o benchmark de alto nível T completo. Não foi preciso construir procedimentos em linguagem de terceira geração para executar as transações.

O COPPEREL possui desempenho razoavelmente semelhante em todas as transações. Por exemplo, na base de dados em teste, o tempo de operação foi inferior a 8 (oito) segundos. Assim, conclui-se que seu desempenho é uniforme ao longo da execução de todas as operações relacionais.

CAPÍTULO VII

CONCLUSÃO

Não existe benchmark perfeito, que possa ser utilizado para avaliar o desempenho de todos os SGBD's em todas as instalações. Conforme foi dito anteriormente, os resultados de desempenho gerados por um benchmark são dependentes da aplicação, recursos e ambiente operacional utilizados. Assim, deve-se utilizá-lo para obter o "melhor" SGBD dentro das necessidades, contexto e escopo da instalação em teste.

Não há vecc ita para um benchmark bem sueedido. Não se con;eguiu dasenvolver uma fóri ula capie de furicionai-, perfeitamente, em todos os SGBD's, instalações e ambientes operi ioi ais. Crntudo, existem banchmarks sufic ieritemente rigorosos para garantir algum grau de sucesso na maioria doa produtos anali;ados. Adicionalmenta, deve-se mencionar que bcnchmavks permariecem como a única mancira prática de estimar o desempenho de SGBD's.

Novos benchmarks estão sendo projetados e construídos [She190] para gerar resultados de desempenho mais precisos e seguros, e que atendam os objetivos de cada ambiente computacional em uso. Aplicações de tempo real requerem recursos precisos, seguros e confiáveis, e necessitam avaliá-los antes do uso. Para tanto, é necessário que existam benchmarks para avaliar tais produtos e verificar se atendem os requisitos envolvidos na instalação. O objetivo principal é obter o "melhor" para o fim desejado.

Não existe um SGBD ideal para executar todos os tipos de aplicações. Geralmente, sua concepção objetiva o desenvolvimento de aplicações em determinadas áreas. Uma análise da funcionalidade do mesmo permite identificar características, recursos e objetivos envolvidos.

Para tanto, nas empresas com atuação em muitas áreas, é preciso testá-los em todas, para observar os desempenhos. O desenvolvimento de várias aplicações e, portanto, vários benchmarks para realizar os testes, é uma tarefa de custo elevado. Uma solução seria determinar os requisitos e objetivos de cada área, construir e implementar uma aplicação (benchmark) única e genérica que incorpore todas estas características, executá-la nos diversos SGBD's que devem ser analisados, analisar os resultados de desempenho gerados pelo benchmark, e escolher o "melhor" SGBD para a instalação em teste.

As pesquisas sobre os SGBD's disponíveis para o mercado de microinformática e o contato com fornecedores permitiram identificar tendências e evoluções na área de banco de dados no Brasil e no mundo. Pode-se analisar produtos nacionais e estrangeiros no mesmo ambiente em relação a parâmetros estáticos e dinâmicos.

Pode-se concluir, também, que existem poucos SGBD's no mercado de microcomputadores que sejam totalmente nacionais. A maioria dos produtos analisados nesta tese é estrangeiro e, representado e comercializado no Brasil por alguma empresa brasileira de informática. O COPPEREL é uma exceção, pois é um SGBD totalmente nacional.

Pode-se concluir, também, que a maioria dos SGBD's que se dizem relacionais, não o são. Alguns possuem modelos próprios, outros possuem características muito similares ao modelo hierárquico e outros ao modelo em redes. Segundo [Codd85], a grande aceitação dos produtos relacionais no mercado, faz com que produtos sejam apresentados como tal ou que algumas características sejam adicionadas ao SGBD de forma a aproximá-lo do modelo relacional, mesmo que este não atenda os requisitos mínimos necessários. Estas informações podem ser comprovadas pelos produtos analisados nesta tese, que em sua maioria não são totalmente relacionais, exceto o COPPEREL que é 100% relacional.

A metodologia ("benchmark") desenvolvida para avaliação do desempenho de SGBD's foi testada em sete produtos selecionados no mercado de microcomputadores. Contudo, esta pode ser usada para avaliar outros SGBD's. Logo, deve-se implementá-la usando os comandos da linguagem de definição/manipulação de dados do produto em teste. O mecanismo de avaliação e os parâmetros de comparação são os mesmos.

A implementação e execução dos benchmarks permitiu identificar características e deficiências em cada SGBD. Não há um produto que seja o "melhor" em todos os parâmetros estáticos e dinâmicos. Deve-se acrescentar que alguns produtos não dispõem de recursos que permitam implementar e analisar alguns parâmetros.

O COPPEREL e ORACLE foram os SGBD's que permitiram a implementação, execução e análise de todos os benchmarks de baixo nível. Enquanto o PARADOX foi aquele cuja linguagem

de definição e manipulação de dados possui menor poder relacional, permitindo a implementação e teste de somente 6 (seis) transações Ti (benchmarks de baixo nível). Apenas 5 (cinco) Ti puderam ser testadas em todos os produtos.

Os resultados de desempenho gerados pelo benchmark de alto nível T não puderam ser usados para analisar os SGBD's. Os valores obtidos estavam distorcidos e sem significado pelo fato de algumas transações não poderem ser implementadas em alguns produtos. Assim, gerou-se outro benchmark de alto nível com escopo limitado, contendo apenas o subconjunto máximo das transações que podiam ser implementadas em todos os SGBD's. Os resultados foram significativos e permitiram gerar comclusões sobre o desempenho dos produtos.

Em ambientes e SGBD's cujo escopo e recursos são limitados, o benchmark de alto nível pode gerar resultados de desempenho não significativos, distorcidos ou incorretos. Não há um ambiente ideal onde todos os benchmarks funcionem com sucesso. Deve-se analisar cuidadosamente os resultados de forma a validá-los corretamente.

Quanto aos parâmetros dinâmicos, concluiu-se que o COPPEREL possui desempenho muito bom quanto a implementação das operações relacionais. Os tempos obtidos são, na maioria das transações, melhores que os dos outros produtos. Assim, não foram propostas melhorias para o SGBD neste aspecto.

A análise funcional dos SGBD's permitiu identificar deficiências no COPPEREL em relação a alguns parâmetros estáticos e propor melhorias para o mesmo. Busca-se

otimizá-lo incorporando recursos que permitam igualá-lo aos melhores produtos do mercado.

Finalmente, deve-se concluir que um SGBD gerado como trabalho de fim de curso em uma das disciplinas do curso de informática da U.F.R.J., sem fins comerciais, possui desempenho muito bom e funcionalidade relativamente boa. Não encontrou-se um produto que fosse muito superior ao COPPEREL. Portanto, após a implementação das otimizações, este estaria em condições de competir com os melhores SGBD's relacionais no mercado de informática.

Em futuros trabalhos, poderia-se extender a amostra de transações, extender o ambiente para multiusuário, gerar os resultados de desempenho com versões mais atualizadas dos SGBD's apesar desta versão do COPPEREL estar em operação desde 1983 e ter sido comparada com sucesso a versões de produtos lançadas "hoje" (aproximadamente 1990), analisar em detalhes o COPPEREL em comparação com apenas um SGBD (por ex: Oracle que possui uma arquitetura similar), e implementar as alterações propostas para o COPPEREL.

REFERÊNCIAS BIBLIOGRÁFICAS

[Aaro86a] Aarons, Richard, "Q&A: A Database with a Smart Interface", PC Magazine V5, N5, Março 1986.

[Aaro86b] Aarons, Richard, "A Data Manager With Flexible Designs", PC Tech Journal V4, N6, Junho 1986.

[Abur88] Aburto, Alfred A. Jr., "Problems and Pitfalls, What's Wrong with the old benchmarks programs?", Byte, Junho 1988.

[AhnI86] Ahn, I. & Snodgrass, R., "Performance Evaluation of a Temporal Database Management System", Proceedings of the 1986 ACM Sigmod International Conference on Management of Data, Washington, Maio 1986.

[Amad87] Amadeo, J'aime, "Oracle for IBM PC/MS-DOS - Installation and User's Guide", Oracle Corporation, Belmont, California, Agosto 1987.

[Ande89] Anderson, T. L. & Berre, A. J. & Mallison, M. & Porter, H. & Schneider, B. , "The Tektronix HyperModel Benchmark Specification", Technical Report No. 89-05, Computer Research Laboratory, Tektronix Laboratories, U.S.A., Agosto 1989.

[Andr85] Anderson, Julie, "Evaluating Data Managers as Development Tools", PC Tech Journal, Agosto 1985.

[Anon88] Anon et all, "A Measure of Transaction Processing Power", Tandem Technical Report TR 85.2 -

February 1985, in "Reading in Database Systems", Michael Stonebraker, Morgan Kanfmann Publishers Incorporation, Califórnia, 1988.

[Ansa87a] "Paradox - A Quick Guide to Paradox for dBASE Users", Release 2.0, Ansa Software, Borland Company, 1987.

[Ansa87b] "Paradox - A Quick Guide to Paradox for Lotus Users", Release 2.0, Ansa Software, Borland Company, 1987.

[Ansa87c] "Introdução ao Paradox", Release 2.0, Ansa Software, Borland Company, 1987.

[Ansa88a] "Paradox Personal Programmer Guide", Release 2.0, Ansa Software, Borland Company, 1988.

[Ansa88b] "Paradox User's Guide", Release 2.0, Ansa Software, Borland Company, 1988.

[Ansa88c] "Paradox Quick Reference Guide", Release 2.0, Ansa Software, Borland Company, 1988.

[Ansa88d] "Paradox PAL Quick Reference Guide", Release 2.0, Ansa Software, Borland Company, 1988.

[Ansa88e] "The PAL User's Guide - The Paradox Application Language", Release 2.0, Ansa Software, Borland Company, 1988.

[Ansa89] "Paradox - The Complete Database", Release 2.0, Prospecto de demostração do produto, Ansa Software, Borland Company, 1989.

[Apik88] Apiki, Steve & Diehl, Stanford, "Benchmarks at a Glance", Byte, Dezembro 1988.

[Bada79] Badal, D. Z. & Popek, G. J., "Cost and Performance Analysis of Semantic Integrity Validation Methods", Proceedings of the 1979 ACM Sigmod International Conference on Management of Data, Boston, Massachusetts, Maio/Junho 1979.

[Badg87] Badgett, Tom, "Where is it? Searching Through Files With Database Software", PC Magazine V6, N18, Outubro 1987.

[Bara86] Baran, Nicholas M., "A Most Ingenious Paradox", PC World V4, N3, Março 1986.

[Bara88] Baran, Nicholas, "DBASE IV: A Paradox Killer?", Byte V13, N4, Abril 1988.

[Barb86] Barbosa, Sergio, "Dataflex: Interativo e Multiusuário", PC Mundo, N7, Fevereiro 1986.

[Bark89] Baran, Nick, "Superbase 4", Byte, Março 1989.

[Barn88] Barney, Douglas, "Users Wary of buggy dBASE IV", Computer World V22, N46, Novembro 1988.

[Bein88] Beinhorn, "Q&A 3.0", PC World V6, N7, Julho 1988.

[Bitt83] Bitton, D. R DeWitt, D. J. R Turbyfill, C., "Benchmarking Database Systems: A Systematic Approach", Proceedings of the 1983 Very Large Database Conference, Florence, Italy, Outubro/Novembro 1983.

[Bitt88] Bitton, D. & Turbyfill, C., "A Retrospective on the Wisconsin Benchmark", in "Reading in Database Systems", Michael Stonebraker, Morgan Kanfmann Publishers Incorporation, Califórnia, 1988.

[Blum88] Blum, H. & Gonçalves, L. & Rossatto, M. A., "Proposta de Desenvolvimento de um Protótipo de Sistema de Gerência de Banco de Dados Orientados a Objetos", Relatório Técnico ES-218/89, COPPE, U.F.R.J., Rio de Janeiro, Dezembro 1988.

[Blum89a] Blum, H. & Gonçalves, L. & Rossatto, M. A. & Mattoso, M., "CPRELOBJ - Um Protótipo de Sistema de Gerência de Banco de Dados Orientados a Objetos", Proceedings of the IX International Conference of the Chilean Computer Science Society & XV Latin American Conference on Informatics, Santiago, Chile, Junho 1989.

[Blum89b] Blum, H. & Gonçalves, L. & Rossatto, M. A., "Implementação de um Protótipo de Sistema de Gerência de Banco de Dados Orientados a Objetos", Relatório Técnico ES-219/89, COPPE/Sistemas, U.F.R.J., Rio de Janeiro, Março 1989.

[Bora84] Boral, H. et all, "A Methodology for Database System Performance Evaluation", Proceedings of the 1984 ACM Sigmod International Conference on the Management of Data, Boston, Junho 1984.

[Bora86] Boral, H. & Brige, R. & DeWitt, D. J. & Gawlick, D. & Hawthorn, P., "Database System Performance

Measurement", Proceedings of the 1986 ACM Sigmod International Conference on the Management of Data, Washington, Maio 1986.

[Bor189] "Paradox - The Complete Database", Version 2.0, Borland International, 1989.

[Borr88] Borr, A. J. & Putzolu, F., "High Performance SQL Through Low-Level System Integration", Proceedings of the 1988 ACM Sigmod International Conference on the Management of Data, Chicago, Illinois, Junho 1988.

[Bott90] Bott, Ed, "R:base Loosens Up - Easy querying and pulldown menus bring R:base release 3.0 face-to-face with Paradox", PC World V8, N4, Abril 1990.

[Bowm86] Bowman, V., "A Data Manager That Synthesizes Text and Data", PC Tech Journal V4, N7, Julho 1986.

[Bric77] Brice, R. S. & Sherman, S. W., "An Extension of the Performance os a Database Manager in a Virtual Memory System Using Partially Locked Virtual Buffers", ACM Transactions on Database Systems V2, N2, Junho 1977.

[Brow86a] Browning, Dave, "A Data Manager: The Evolving Standard", PC Tech Journal V4, N5, Maio 1986.

[Brow86b] Browning, Dave, "A Data Manager for End-User Development", PC Tech Journal V4, N9, Setembro 1986.

[Brow87] Browning, D. & Blasdel, H., "Managing Databases, Mainframe Style", PC Tech Journal V5, N12, Dezembro 1987.

[Byte87] "High-Tech Horsepower", by the Byte Editorial Staff, Byte, Julho 1987.

[Byte89] "The Data File", Byte V14, N9, Setembro 1989.

[Camp88] Campos, Alda, "Como o Oracle está sendo usado no Brasil", Datanews, Abril 1988.

[Card75] Cardenas, Alfonso F., "Analysis and Performance of Inverted Data Base Structures", Communications of the ACM V18, N5, Maio 1975.

[Casa85] Casanova, M. A. & Moura, A. V., "Princípios de Sistemas de Gerência de Bancos de Dados Distribuídos", Editora Campus, 1985.

[Chan85] Chan, A. & Sarin, Sunil, "Improving Availability and Performance of Distributed Database Systems", Database Engineering V8, N1, Março 1985.

[Chap88] Chapel, Hal, "Omnis Quartz 1.13", PC World V6, N7, Julho 1988.

[Chri84] Christodoulakis, S., "Implications of certain assumptions in Database Performance Evaluation", ACM Transactions on Database Systems, New York V9, N2, Junho 1984.

[Chrs85] Christian, Chris, "A Data Manager for Diverse Environments", PC Tech Journal, Agosto 1985.

[Cobb87] Cobb, Stephen, "Rbase System V", Byte V12, N4, Abril 1987.

[Codd79] Codd, E. F., "Extending the Database Relational Model to Capture More Meaning", ACM Transactions on Database Systems V4, N4, Dezembro 1979.

[Codd85] Codd, E. F., "An Evaluation Scheme for Database Management Systems that are claimed to be Relational", CW Communications Inc., ComputerWorld, 1985.

[Cole87] Coleman, David & Jackson, Gregg, "DeWitt Benchmark: ORACLE vs. INGRES on Sun", Technical Report No. 20252 - 1187, Oracle Corporation, Belmont, California, Novembro 1987.

[Comp89a] "Oracle, DBMS Relacional", Compucenter, Datanews, Março 1989.

[Comp89b] "Introduction to SQL*Forms", Curso de Oracle, Compucenter, 1989.

[Comp89c] "Introdução ao Desenvolvimento de Aplicações em Oracle", Curso de Oracle, Compucenter, 1989.

[Comp89d] "O padrão em Banco de Dados", CI-Compucenter Informática, Setembro 1989.

[Comp89e] "ORACLE", Informativo da CI-Compucenter Informática, Ano I, N1. Setembro/Outubro 1989.

[Copp88] "Normas para elaboração, apresentação gráfica e defesa de teses de M.Sc. e D.Sc.", COPPE/UFRJ, Rio de Janeiro, Junho 1988.

[Corr88] Corrigan, Patrick H., "What to Look for in a LAN Database", Micro/Systems, Setembro 1988.

[Cose86a] Cosentino, Laercio J. L., "dBase III - Interativo", Editora Atlas S. A., 1986.

[Cose86b] Cosentino, Laercio J. L., "dBase III - Programado", Editora Atlas S. A., 1986.

[Crab87] Crabb, Don, "Paradox Database Offers Features, Reliability", Info World V9, Issue 33, Agosto 1987.

[Curn76] Curnow, H. J. & Wichmann, B. A., "A Synthetic Benchmark", The Computer Journal V19, N1, Fevereiro 1976.

[Curo86] Curotto, A. A. R. & Werner, C. M. L., "Pesquisa de Sistemas Gerenciadores de Banco de Dados em PCs", Petrobrás, Novembro 1986.

[Data85a] "Iniciando o DBASE III Plus", Ashton-Tate, Datalógica, 1985.

[Data85b] "Gerador de Aplicativos DBASE III Plus", Ashton-Tate, Datalógica, 1985.

[Data86a] "DBASE III PLUS - Aprendendo o dBase III Plus", Versão 1.0, 1a. edição, Ashton-Tate, Datalógica, Julho 1986.

[Data86b] "DBASE III PLUS - Programando com o dBase III Plus", Versão 1.0, 1a. edição, Ashton-Tate, Datalógica, Julho 1986.

[Data86c] "DBASE III PLUS - Usando o dBase III Plus", Versão 1.0, 1a. edição, Ashton-Tate, Datalógica, Julho 1986.

[Data86d] "DBASE III PLUS - Colocando o dBase III Plus em Redes", Versão 1.0, 1a. edição, Ashton-Tate, Datalógica, Julho 1986.

[Data86e] "DBASE III PLUS - RUNTIME+", Versão 1.0, 1a. edição, Ashton-Tate, Datalógica, Julho 1986.

[Data89a] "FRONT RUNNER - O passaporte para o dBASE IV", Prospecto de apresentação do produto, Ashton-Tate, Datalógica, 1989.

[Data89b] "DBASE IV", Prospecto de apresentação do produto, Ashton-Tate, Datalógica, 1989.

[Data89c] "DBASE IV - O novo padrão em Banco de Dados", Prospecto de apresentação do produto, Ashton-Tate, Datalógica, 1989.

[Data89d] "DBASE IV - News", Ashton-Tate, Datalógica, 1989.

[Date83] Date, C. J., "An Introduction to Database Systems", Volume 2, Addison-Wesley Publishing Company, 1983.

[Date87] Date, C. J., "An Introduction to Database Systems", Volume 1, Fourth Edition, Addison-Wesley Publishing Company, 1987.

[Davi89] Davis, Ralph, "Sharing the Wealth", Byte V14, N9, September 1989.

[Daws89] Dawson, Joseph, "A Family of Models", Byte V14, N9, Setembro 1989.

[Desp87] Desposito, Joe, "File Managers: Get a Face-lift", PC Magazine V6, N2, Janeiro 1987.

[DeMa86] DeMaria, R., "Paradox 1.1", Byte, Setembro 1986.

[Demu85] Demurjian, S. & Hsiao, D. K., "Benchmarking Database Systems in Multiple Backend Configurations, Database Engineering V8, N1, Março 1985.

[Derf86] Derfler, Frank J. Jr., "Paradox: A Database Manager With a Familiar Face", PC Magazine V5, N1, Janeiro 1986.

[Derf88] Derfler, Frank J. Jr., "Making Connections: Programmable Databases for LANs", PC Magazine V7, N19, Novembro 1988.

[Dewi81] DeWitt, David J. & Hawthorn, P., "A Performance Evaluation of Data Base Machine Architectures", Proceedings of the 1981 Very Large Database Conference, Cannes, France, Setembro 1981

[DeWi85] DeWitt, David J., "Benchmarking Database Systems: Past efforts and Future Directions", Database Engineering V8, N1, Março 1985.

[DeWi88] DeWitt, David J. & Ghandeharizadeh, S. & Schneider, D., "A Performance Analysis of the Gamma Database Machine", Proceedings of the 1988 ACM Sigmod International Conference on Management of Data, Chicago, Illinois, Junho 1988.

[Dick86a] Dickinson, John, "Project Data Base II: Programmable Relational Databases", PC Magazine V5, N12, Junho 1986.

[Dick86b] Dickinson, J., "Project Data Base II: Relational Databases", PC Magazine V5, N12, Junho 1986.

[Dill85] Dillenbeck, Bruce, "Fighting fire with Technology", Byte, Outubro 1985.

[Dimm86] Dimmick, S., "The Oracle Database Administrator's Guide", Oracle Corporation, Agosto 1986.

[Dowg88] Dowgiallo, Edward, "An Introduction to SQL", Micro/Systems, Setembro 1988.

[Duhl88] Duhl, J. & Damon, C., "A Performance Comparison of Object and Relational Databases Using the Sun Benchmark", Proceedings of the 1988 OOPSLA Object Oriented Programming Systems, Languages and Applications Conference, Setembro 1988.

[Edel89a] Edelstein, Herb A., "In Front of the Server", PC Tech Journal V7, N3, Março 1989.

[Edel89b] Edelstein, Herb A., "OS/2 Meets SQL", PC Tech Journal V7, N2, Setembro 1989.

[Edel90] Edelstein, Herb A., "Auditing the Auditors", DBMS V3, N3, Março 1990.

[Edwa85] Edwards, Jon R. & Hartwig, Glenn, "Benchmarking the Clones", Byte, Especial Issue Inside the IBM PCs, Fall 1985.

[Eich85] Eich, M., "Transaction Oriented Performance Analysis of Database Machines", Database Engineering V8, N1, Março 1985.

[Eise86] Eisenberg, Robert, "A Plus for DBASE III", PC World, Outubro 1986.

[Elet87] "DBASE III - Guardando as informações", Manual M050, Eletrobrás, 1986.

[Falk87] Falkner, Mike, "Crossing the Great Divide", PC Magazine V6, N22, Dezembro 1987.

[Fast89] Fastie, Will, "Paradox Made Better", PC Tech Journal V7, N2, Fevereiro 1989.

[Fedo84] Fedorowicz, J., "Database Evaluation Using Multiple Regression Techniques", Proceedings of the 1984 ACM Sigmod International Conference on Management of Data, Boston, Junho 1984.

[Fedo87] Fedorowicz, J., "Database Performance Evaluation in an Indexed File Environment", ACM Transactions on Database Systems V12, N1, Março 1987.

[Fink87] Finkelstein, R., "Lingua Franca for Databases", PC Tech Journal V5, N12, Dezembro 1987.

[Fink88] Finkelstein, R. & Pascal, F., "SQL Database Management Systems", Byte V13, N1, Janeiro 1988.

[Fink89] Finkelstein, R., "A Database Server Odyssey", PC Tech Journal V7, N3, Março 1989.

[Fink90] Finkelstein, R., "The Unix Advantage", DBMS V3, N3, Março 1990.

[Flem86] Fleming, Philip J. & Wallace, John J., "How not to lie with Statistics: The Correct Way to Summarize Benchmark Results", Communications of the ACM V29, N3, Março 1986.

[FoxD89] Fox, David L., "Benchmarking C Statements", Dr. Dobb's Journal, Canada, Fevereiro 1989.

[FoxR88] Fox, Ron, "Why MIPS Are Meaningless, Component Benchmarks tell you about subsystems, not about the system as a whole", Byte, Junho 1988.

[Fran88] Franklin, Curtis Jr., "SQL-based Database Managers", Byte, Janeiro 1988.

[Gaba88] Gabaldon, Diana, "Database Management via 1-2-3", Byte, Maio 1988.

[Ghos76] Ghosh, Sakti P. & Tuel Jr., William G., "A Design of an Experiment to Model Data Base System Performance", IEEE Transactions on Software Engineering Vol SE-2, N2, Junho 1976.

[Gilb81] Gilbreath, Jim, "A High-Level Language Benchmark", Byte V6, N9, Setembro 1981.

[Gilb83] Gilbreath, Jim & Gilbreath, Gary, "Eratosthenes Revisited - Once More Througg The Sieve", Byte, Janeiro 1983.

[Gold90] Goldberg, Cheryl J., "Paradox Engine For C Programs", DBMS V3, N3, Março 1990.

[Gonç90] Gonçalves, Laércio Antonio Castelo Branco, "O Sub-Sistema de Reconstrução do SGBD COPPEREL", Tese de Mestrado em Engenharia de Sistemas e Computação, COPPE/Sistemas, U.F.R.J., Rio de Janeiro, Junho 1990.

[Greh87] Grehan, R., "A Closer Look", Byte, Setembro 1987.

[Greh88] Grehan, Richard & Thompson, Tom & Jr., Curtis Franklin & Stewart, George A., "introducing the New Byte Benchmarks", Byte, Junho 1988.

[Hale88] Haleshaw, Richard, "SQL: An Emerging Database Standard for PCs", PC Magazine, Maio 17, 1988.

[Hamm79] Hammer, M. & Niamir, B., "A Heuristic Approach to Attribute Partitioning", Proceedings of the 1979 ACM Sigmod International Conference on Management of Data, Boston, Massachusetts, Maio/Junho 1979.

[Hams88] Hanson, E., "Processing Queries Against Database Procedures: A Performance Analysis", Proceedings of the 1988 ACM Sigmod International Conference on Management of Data, Chicago, Junho 1988.

[Hart85] Hart, Glenn A., "A Smorgasbord of Fox & Geller Enhancements for DBASE III", PC Magazine V4, N23, Novembro 1985.

[Hart86] Hart, Glenn A,,, "Adding Speed and Functions to DBASE", PC Magazine V5, N12, Junho 1986.

[Hart87a] Hart, Glenn A., "DBASE: Power Tools", PC Magazine V6, N13, Julho 1987.

[Hart87b] Hart, Glenn A., "Faster FoxBASE Allows Custom Functions", PC Magazine V6, N20, Novembro 1987.

[Hart87c] Hart, Glenn A., "DBASE Report Writers: A Better Way", PC Magazine V6, N21, Dezembro 1987.

[Hash81] Hashin, R. L. & Lorie, R. A., "On Extending the Functions of a Relational Database System", IBM Research Laboratory, San José, California, 1981.

[Hawt79] Hawthorn, Paula B. & Stonebraker, M., "Performance Analysis of a Relational Data Base Management System", Proceedings of the 1979 ACM Sigmod International Conference on Management of Data, Boston, Massachusetts, Maio/Junho 1979.

[Hawt82] Hawthorn, Paula B. & DeWitt, David J., "Performance Analysis of Alternative Database Machine Architectures", IEEE Transactions on Software Engineering Vol SE-8, N1, Janeiro 1982.

[Hawt85] Hawthorn, Paula B., "Variations on a Benchmark", Database Engineering V8, N1, Março 1985.

[Hel185] Helliwell, John, "The World According to Zim", PC Magazine V4, N18, Setembro 1985.

[IBM_85] "Banco de Dados Ralacional num ambiente de centro de informações", Informação IBM, N23, Setembro 1985.

[IBM_87] "IBM Database 2 Performance Report", International Technical Support Center, Document Number GG24-3146-00, San Jose, California, U.S.A., Fevereiro 1987.

[Info87] "Programação facilitada, o gerenciador de Banco de Dados da Iesa suprime os comandos e inova com o uso de funções", Info, Dezembro 1987.

[Info88a] "Quatro produtos em teste: Clipper, DataEase, Dataflex e Open Access II, Vantagens e Desvantagens", Info V6, N66, Julho 88.

[Info88b] "O Representante Nacional: os testes do Zim e do Dialog continuam a série sobre SGBD's", Info, Outubro 1988.

[Info88c] "Para quem sabe o que quer: O Oracle 5.1 é um SGBD de vastos recursos se explorado adequadamente", Info, Dezembro 1988.

[Info88d] "Parasiox, o gi-ande vencedor-: Os testes do DBASE III Plus e IV, PC/Focus e Paradox", Info, Agosto 1988.

[Info88e] "Com altos e baixos, agil nas operações simples, o Dataflex não responde a algumas funções mais elaboradas", Info, Fevereiro 1988.

[Info88f] "Contra os preconceitos, o Dialog Plus chega com comandos em português e faz do compilador "dois em um" sua grande atração", Info, Janeiro 1988.

[Info89] "Rapidez, a força do DBASE IV", Info, Janeiro 1989.

[Inte___] "Comparação Técnica Paradox X dBase III/Plus", Relatório Técnico, Intercorp do Brasil Ltda.

[Inte88] "The PAL Translator", Versão 2.0, Intercorp do Brasil Ltda., Janeiro 11, 1988.

[Inte89a] "ADD-INS: Poderosas Ferramentas que integram e completam o 1-2-3", Prospecto de apresentação do produto, Intercorp do Brasil Ltda., 1989.

[Inte89b] "Paradox - O mais completo Gerenciador de Banco de Dados Relacional", Prospecto de apresentação do produto, Intercorp do Brasil Ltda., 1989.

[Intr89a] "Dataflex", Íntercomp Journal, Interamericana de Computação Ltda., 1989.

[Intr89b] "Dataflex", Prospecto de apresentação do produto, Interamericana de Computação Ltda., 1989.

[Jacq85] Jacques, Jeffrey M., "DB Master for the Macintosh", Byte, Setembro 1985.

[Jend87] Jendricks, John, "SQL*Report", Oracle Corporation, 1987.

[John90] Johnston, Elizabrith S., "Paradox Tables", DBMS V3, N3, Março 1990.

[Juni86] Junior, Ataliba Faria da Silva, "Rapidez no DBASE", PC Mundo, Maio 1986.

[Kahn84] Kahn, S., "An Overview of Three Relational Database Products", IBM Systems Journal V23, N2, 1984.

[Karp85] Karpinski, Richard, "Paranoia: A Floating-point Benchmark", Byte, Fevereiro 1985.

[KarR89] Kar, Rabindra P. & Porter, Kent, "Rhealstone: A Real Time Benchmarking Proposal", Dr. Dobb's Journal, Canada, Fevereiro 1989.

[KMCo89a] "Zim - A Sustentável Leveza do Software", Prospecto de apresentação do produto, KM Consultoria e Editores Associados Ltda., 1989.

[KMCo89b] "Zim - Ferramenta para desenvolvimento de Aplicações", Prospecto de apresentação do produto, KM Consultoria e Editores Associados Ltda., 1989.

[Knut73] Knuth, D. E., "The Art of Computer - Programming, Searching and Sorting", Volume 3, Addison Wesley Publishing Company, 1973.

[Kraj84] Krajewski, Rich, "Database Types", Byte, Outubro 1984.

[Kras86a] Krasnoff, Barbara & Dickinson, John, "Project Data Base II", PC Magazine V5, N12, Junho 1986.

[Kras86b] Krasnoff, B., "Project Data Base II: Flat-file Databases", PC Magazine V5, N14, Agosto 1986.

[Lent89] Lent, A. F. & Rubel, M., "DBASE IV: Setting the New Standard?", Byte V14, N1, Janeiro 1989.

[Luzi87] Luzio, J. M., "Compiladores DBASE", PC Mundo, Junho 1987.

[Maga81] Magalhães, Geovane C., "Improving the Performance of Data Base Systems", Ph. D. Thesis, Technical Report CSRG-138, Department of Computer Science, University of Toronto, Canada, Dezembro 1981.

[Magr88] Magri, João Alexandre, "Dialog Plus/C Interativo - Sistema para Gerenciamento de Banco de Dados para Microcomputadores", Editora Atlas, 1988.

[Mall88] Malloy, Rich, "Excel Extraordinaire", Byte, Março 1988.

[Mars86] Martins, A. S. & Uchoa, A., "Uma Versão mais veloz e amigável", PC Mundo, Outubro 1986.

[Mart86] Martin, Daniel, "Advanced Database Techniques", MIT Press Serves in Information Systems, London, England, 1986.

[Mato89] Matos, Victor M. & Ialics, Paul J., "An Experimental Analysis of the Performance of Fourth Generation Tools on PCs", Communications of the ACM V32, N11, Novembro 1989.

[Matt85a] Mattoso, Marta L. Q. & Zahimi, B. M., "Manual do Administrador das Bases de Dados do SGBD Copperel", Relatório Técnico ES-68/85, COPPE/Sistemas, U.F.R.J., Rio de Janeiro, 1985.

[Matt85b] Mattoso, Marta L. Q. & Zahimi, B. M., "Manual de Instalação do SGBD Copperel", Relatório Técnico ES-63/85, COPPE/Sistemas, U.F.R.J., Rio de Janeiro, 1985.

[Matt85c] Mattoso, Marta L. Q. R Zahimi, B. ti,, R Prazeres, Manoel % Souza, Jano M. de & DeSimone, Estevari, "Lopercl: 11ma Liriguagem de Quarta Geração e sua Utilização em Ambiente de um Sistema de Gerência de Banco de Dados", Anais da Convencion Informatica Latina, Palau de Congressos de Montjuic, Barcelona, Espanha, Abril, 1985.

[Matt85d] Mattoso, Marta L. Q. & Zahimi, B. M., "Manual de Instalação do SGBD Copperel", Relatório Técnico ES-49/85, COPPE/Sistemas, U.F.R.J., Rio de Janeiro, 1985.

[Matt87] Mattoso, Marta Lima de Queirós, "Incorporação de Ferramentas de Apoio aos Usuários de SGBD Copperel", Tese de Mestrado em Engenharia de

Sistemas e Computação, COPPE/Sistemas, U.F.R.J., Rio de Janeiro, Maio 1987.

[Mede88] Medeiros, Gefferson de & Santos, Luciano dos, "Desempenho no Relacional: uma Experiência com o SQL/DS", Datanews, Abril 1988.

[Mias88] Miastkowski, Stan, "Paradox Takes on OS/2", Byte V13, N8, Agosto 1988.

[Mias89] Miastkowski, Stan & Nick, B., "Paradox3 Neither Enigma nor Riddle", Byte V14, N2, Fevereiro 89.

[Micr86a] "MicroSoft Rbase System - Learning Guide", MicroSoft Corporation, 1986.

[Micr86b] "MicroSoft Rbase System - Building Applications", MicroSoft Corporation, 1986.

[Micr86c] "MicroSoft Rbase System - User's Manual", MicroSoft Corporation, 1986.

[Micr86d] "MicroSoft Rbase System - Rbase Command Dictionary", MicroSoft Corporation, 1986.

[Micr86e] "MicroSoft Rbase System - The Database Management Solution", MicroSoft Corporation, 1986.

[Mout88a] Moutinho, F. M., "SGBD's Relacionais em micros: quanto vale este esforço", Datanews, Maio 1988.

[Mout88b] Moutinho, F. M., "Oracle em micros: Eficiência, Segurança e Compatibilidade", Datanews, Novembro 1988.

[Mudi88] Mudie, Mary W., "DB2 Implementation Primer", Enterprise and Application Enabling Systems Support Center, Dallas Systems Center, Technical Bulletin Number 0066-3117-00, Dezembro 1988.

[Name89] Name, Mark L. Van & Catchings, Bill, "Serving Up Data", Byte V14, N9, Setembro 1989.

[Nich88] Nicholls, Bill, "That 'B' Word! What it is, Where its going, and Why we subject ourselves to it", Byte, Junho 1988.

[Nogu86] Nogueira, José Luiz da Silva, "Rapidez no trato com arquivos", PC Mundo, Agosto 1986.

[O'Ha84] "The Sensible Solution - Language, Installation, Reference", O'Hanlon Computer Systems, 1984.

[Olym88] Olympia, P. L., "Database Queries: Report Writing in DBASE IV", Micro/systems, Setembro 1988.

[OraB89a] "Oracle: Estabelecendo o Padrão Relacional", Informativo Técnico 0101-0989, Oracle do Brasil, Setembro 1989.

[OraB89b] "O SGBD Relacional ORACLE: Desempenho, Poder e Integridade", Informativo Técnico 0201-0989, Oracle do Brasil, Setembro 1989.

[OraB89c] "SQL*STAR TM: Tecnologia de Amanhã Disponível Hoje", Informativo Técnico 0301-0989, Oracle do Brasil, Setembro 1989.

[OraB89d] "Ferramentas de Produtividade para o Profissional de PD", Informativo Técnico 0401-0989, Oracle do Brasil, Setembro 1989.

[OraB89e] "Poder e Simplicidade para o Usuário Final", Informativo Técnico 0501-0989, Oracle do Brasil, Setembro 1989.

[OraB89f] "Soluções Personalizadas, sem programação", Informativo Técnico 0601-0989, Oracle do Brasil, Setembro 1989.

[OraB89g] "Engenharia de Sistemas Auxiliada por computador (CASE)", Informativo Técnico 0701-0989, Oracle do Brasil, Setembro 1989.

[OraC___] "Oracle - Features and Characteristics of Commercial Offerings", REV 26-01-16, Oracle Corporation, Auerbach Publishers Inc, Menlo Park.

[OraC84a] "Oracle SQL/UFI Reference Guide", Oracle Corporation, California, 1984.

[OraC84b] "Oracle IAF Terminal Operator's Guide", Oracle Corporation, California, Outubro 1984.

[OraC84c] "Oracle IAF Application Designer's Guide", Oracle Corporation, California, Outubro 1984.

[OraC84d] "Oracle RPT Report Generator User's Guide", Oracle Corporation, California, Outubro 1984.

[OraC84e] "Oracle Database Administrator's Guide", Oracle Corporation, California, Outubro 1984.

[OraC84f] "Oracle Installation Guide", Oracle Corporation, California, 1984.

[OraC85] "Introduction to SQL", Oracle Corporation, 1985.

[OraC86] "Introduction to SQL*Forms", Version 2.3, Oracle Corporation, 1986.

[OraC87] "SQL*Forms - Designer's Tutorial", Version 2.0, Oracle Corporation, Julho 1987.

[OraC88] "Annual Report", Oracle Corporation, 1988.

[OraC89] "Products and Services Overview", Oracle Corporation, Janeiro 1989.

[Ozka77] Ozkarahan, E. A. & Schuster, S. A. & Sevcik, K. C., "A Performance Evaluation of a Relational Associative Processor", ACM Transactions on Database Systems V2, N2, Junho 1977.

[Pale85] Palermo, Luigino Italo & Mattoso, Marta L. Q. & Zakimi, Beatriz M., "Proposta de uma Versão Multiusuário para o SGBD COPPEREL", Relatório Técnico ES-79/85, COPPE/Sistemas, U.F.R.J., Rio de Janeiro, Agosto 1985.

[Palm87] Palmer, Scott D., "Dataease Relational Database Powerful But Easy to Learn", Info World V9, Issue 34-35, Agosto 1987.

[Pasc87] Pascal, Fabian, "Relational Power, PC Ease", PC Tech Journal V5, N12, Dezembro 1987.

[Pasc89] Pascal, Fabian, "Database Trends: A Brave New World?", Byte V14, N9, Setembro 1989.

[PCMu88a] "SQL, Chave-mestra dos futuros arquivos", PC Mundo, Julho 1988.

[PCMu88b] "SQL, Altos e baixos nas opções para micros", PC Mundo, Agosto 1988.

[Pere89] Pereira, Rafles Fransino & Magalhães, Geovane Cayres, "Análise de Desempenho de Banco de Dados utilizando uma metodologia baseada em Benchmark Especializado", 4o. Simpósio Bradileiro de Banco de Dados, Campinas, São Paulo, Abril 1989.

[Petr87] Petreley, Nicholas, "Relational Databases - Product Comparison", Info World V9, Issue 34-35, Agosto 1987.

[Phel86] Phelps, Ken, "Os Sofisticados Recursos do Revelation", Info, Setembro 1986.

[Phil23] Phillips, John R., "Multiple Approaches to Multiuser Database Management", PC Magazine V4, N23, Novembro 1985.

[Poor85a] Poor, A., "Microrim's R:Base 5000-A Database that Delivers", PC Magazine V4, N18, Setembro 1985.

[Poor85b] Poor, Alfred, "Cost-Effective Database Management", PC Magazine V4, N21, Outubro 1985.

[Poor87] Poor, Alfred, "Database Power Puts on an Easy Interface", PC Magazine V6, N2, Janeiro 1987.

[Poor89] Poor, Alfred, "DataEase", PC Magazine V8, N1, Janeiro 1989.

[Poun85] Pountain, Dick, "New Database Ideas", Byte, Abril 1985.

[Pres88] Press, Larry, "Benchmarks for LAN Performance Evaluation", Edgar H. Sibley Panel Editor, Communications of the ACM V31, N8, Agosto 1988.

[Pric19] Price, Walter J., "A Benchmark Tutorial", SEEE Micro V9, N5, Outubro 1989.

[Pug186a] Puglia, Vincent, "Testing The Databases, PC Magazine V5, N12, Junho 1986.

[Pug186b] Puglia, Vincent, "TBMS: Database Power Unleashed", PC Magazine V5, N20, Novembro 1986.

[Rag189] Ragland, C. E. Jr., "Data Systems Workload Performance DB2", Enterprise and Application Enabling Systems Support Center, Dallas Systems Center, Technical Bulletin Number 0066-3114-00, Janeiro 1989.

[Rama88a] Ramalho, J. A., "Summer 87, a Versão turbo do Clipper", PC Mundo, Agosto 1988.

[Rama88b] Ramalho, J. A., "O uso de telas padronizadas em um sistema", PC Mundo, Setembro 1988.

[Rama88c] Ramalho, J. A., "Programação com ganhos e vantagens", PC Mundo, Outubro 1988.

[Rama88d] Ramalho, J. A., "DBASE IV, as primeiras impressões", PC Mundo, Novembro 1988.

[Rama88e] Ramalho, J. A., "Aprenda a técnica do menu rotativo", PC Mundo, Dezembro 1988.

[Rama89a] Ramalho, J. A., "DataEase, uma alternativa ao padrão DBASE", PC Mundo, Abril 1989.

[Rama89b] Ramalho, J. A., "Front-Runner, ponte para o dBASE IV", PC Mundo, Agosto 1989.

[Ries77] Ries, David R. & Stonebraker, Michael R., "Effects of Locking Granularity on Database Management System Performance", ACM Transactions on Database Systems V2, N3, Setembro 1977.

[Ries79] Ries, David R. & Stonebraker, Michael R., "Locking Granularity Revisited", ACM Transactions on Database Systems V4, N2, Junho 1979.

[Robe86] Roberts, J., "A Data Manager for the Self Reliant User", PC Tech Journal V4, N10, Outubro 1986.

[Robi89] Robinson, Celeste & King, Steve, "More Power From Paradox", PC World V7, N4, Abril 1989.

[RobJ88] Robie, Jonathan, "Fast Data Access", Byte, Janeiro 1988.

[Rose87a] Rosen, M., "Programmatic Interfaces", Oracle Corporation, 1987.

[Rose87b] Rosen, M., "SQL*Plus", Oracle Corporation, 1987.

[Roti90] Roti, Steve, "In Favor of Xenix - Oracle on Xenix gives you more bang for your operating system buck", DBMS V3, N3, Março 1990.

[Rube87] Rubel, Malcolm C., "Benchmarking DBASE III Plus Compilers", Byte V12, N10, Setembro 1987.

[Rube89] Rubel, Malcolm, "DBASE IV Arrives", Byte V14, N2, Fevereiro 1989.

[Rubn87] Rubenstein, W. B. & Kubicar, M. S. & Cattell, R. G. G., "Benchmarking Simple Database Operations", Proceedings of the 1987 ACM Sigmod International Conference on the Management of Data, San Francisco, California, Maio 1987.

[Rubk87] Rubenking, N. J., "R:base Understands Limited SQL and Runs Faster", PC Magazine V7, N8, Abril 1988.

[Sacc86] Sacco, G. M. & Schkolnick, M., "Buffer Management in Relational Databases Systems", ACM Transaction on Database Systems V11, N4, Dezembro 1986.

[Sach87] Sachs, Jonathan, "SQL*Plus - User's Guide", Version 2.0, Oracle Corporation, Julho 1987.

[Sand88] Sander, E. J., "DataEase 2.5 release 3", PC World V6, N7, Julho 1988.

[Sano89] Santos, Silvia R. F. Rangel dos & Martini, José Guilherme Alves, "Guia para Seleção de Sistemas Gerenciadores de Banco de Dados", Anais do XXII Congresso Nacional de Informática, Sucesu, Parque Anhembi, São Paulo, Brasil, Setembro 1989.

[Sant87a] Santos, Alexandre Carvalho dos, "Registros em Ordem: Criação e Manipulação de um Arquivo de índices em DBASE", PC Mundo, Setembro 1987.

[Sant87b] Santos, Alexandre Carvalho dos, "Fácil de operar até por leigos", PC Mundo, Novembro 1987.

[Schw88] Schwartz, Alan, "FoxBase+/386", PC World V6, N7, Julho 1988.

[Sevc81] Sevcik, K. C., "Data Base System Performance Prediction Using an Analytical Model", Proceedings of the 1981 Very Large Database Conference, Cannes, France, Setembro 1981.

[Seym88a] Seymour, Jim, "Project Data Base 3: Covering all the Bases", PC Magazine V7, N7, Abril 1988.

[Seym88b] Seymour, Jim, "Relational Databases: Taking the Middle Ground", PC Magazine, Abril 1988.

[Seym88c] Seymour, Jim, "Project Database III - Programmable Databases: DBASE and its Challengers", PC Magazine, Maio 1988.

[Sham88] Shammas, Namir Clement, "DBASE MAC Vs MCMAX", Byte V13, N2, Fevereiro 1988.

[Shaw88] Shaw, Richard Hale, "Paradox OS/2: A Good Reason to Buy OS/2", PC Magazine V7, N15, Setembro 1988.

[Shaw89] Shaw, Richard Hale, "Paradox OS/2, Paradox 386", PC Magazine V8, N1, Janeiro 1989.

[Shaw90a] Shaw, Richard Hale, "Quadbase-SQL Brings Real Relational Power to dBase and C", PC Magazine V9, Abril 1990.

[Shaw90b] Shaw, R., H., "Databases: Short-term Efficiency, Lorig-term Productivity For Database Users", Trudy Neuhaus Editor, PC Magazine V9, Abril 1990.

[Shaw90c] Shaw, Richard Hale, "Databases", Trudy Neuhaus Editor, PC Magazine V9, Maio 1990.

[Shel90] Sheldon, Kenneth M., "To Boldly Benchmark", Byte, Abril 1990.

[Shep87] Sheppard, Byron, "High-Performance Software Analysis on the IBM PC", Byte, Janeiro 1987.

[Sher86] Sherman, Stephen W. & Brice, Richard S., "Performance of a Database Manager in a Virtual Memory System", ACM Transactions on Database Systems V1, N4, Dezembro 1976.

[Sirc86] Sircar, Sumit & Dave, Dinesh, "The RelationShip Between Benchmark tests and microcomputer price", Edgar H. Sibley Panel Editor, Communications of the ACM V29, N3, Março 1986.

[Smil87] Smilgiewicz, Gene, "dBase IV Getting a Major Overhaul and Speed Boost", PC Magazine V7, N8, Abril 1988.

[Soft88a] "Dialog Plus/C", Prospecto de apresentação do produto, Soft Consultoria em Processamento de Dados Ltda, 1988.

[Soft88b] "Dialog Plus/C - Manual de Referência", Soft Consultoria em Processamento de Dados Ltda, 1988.

[Soft89a] "Dialog - Catálogo", Soft Consultoria em Processamento de Dados Ltda, 1989.

[Soft89b] "O Estado da Arte do Dialog Plus/C", Soft & Fatos, Informativo da Soft Consultoria em Processamento de Dados Ltda, N11, Ano 2, Agosto 1989.

[Soua88] Souza, Marcio Ferreira de, "Ferramentas para Banco de Dados", Info, Março 1988.

[Souz78] Souza, Jano Moreira de, "Algoritmos de Hashing para Problemas Específicos", Tese de Mestrado em Engenharia de Sistemas e Computação, COPPE/Sistemas, U.F.R.J., Rio de Janeiro, 1978.

[Souz82] Souza, Jano Moreira de & Zakimi, Beatriz & De Simone, Esteven & Prazeres, Manoel José, "Copperel: Sistema de Gerência de Base de Dados da COPPE", Relatório Técnico ES - 17/82, COPPE/Sistemas, U.F.R.J., Rio de Janeiro, 1982.

[Souz88] Souza, Jano Moreira de & Mattoso, Marta Lima de Queiroz, "COPPEREL-PC: A Versão do SGBD COPPEREL para Microcomputadores Tipo PC", Anais do 3o. Simpósio Brasileiro de Banco de Dados, Recife, Pernambuco, Março 1988.

[Spez88] Spezzano, Charles, "Two Mac Databases Go Toe-to-Toe", Byte V13, N6, Junho 1988.

[Spie72] Spiegel, M. R., "Estatística", Coleção Schãum, Tradução Pedro Cosentino, Editora McGraw-Hill do Brasil Ltda, São Paulo-Rio de Janeiro, 1972.

[Star90] Starkey, Jim, "The Next Generation", DBMS V3, N3, Março 1990.

[Ston85] Stonebraker, M., "Tips on Benchmarking Data Base Systems", Database Engineering V8 N1, Março 1985.

[Ston88] Stonebraker, Michael, "Performance And Database Machines", in "Reading in Database Systems", Michael Stonebraker, Morgan Kanfmann Publishers Incorporation, Califórnia, 1988.

[Stre90] Streho, Kevin, "Benchmark Bloodbath", DBMS V3, N3, Março 1990.

[Tand88] "A Benchmark of NonStop SQL on the Debit Credit Transaction", The Tandem Performance Group, Proceedings of the 1988 ACM Sigmod International Conference on Management of Data, Chicago, Illinois, Junho 1988.

[Taur86] Taurion, Cezar, "SGBD's Relacionais", Datanews, Agosto 1986.

[Taur88a] Taurion, Cezar, "Portabilidade, determinando a vantagem do Oracle", Datanews, Abril 1988.

[Taur88b] Taurion, Cezar, "Seleção de SGBD, questão de tempo e investimento", Datanews, Junho 1988.

[Taur88c] Taurion, Cezar, "Até onde os Benchmarks são parâmetros confiáveis", Datanews, Outubro 1988.

[TayY85] Tay, Y. C. & Goodman, N. & Suri, R., "Locking Performance in Centralized Databases", ACM Transactions on Database Systems V10, N4, Dezembro 1985.

[Taze88] Tazelaar, J. M., "Benchmarks: Introduction", Byte, Junho 1988.

[Taze89] Tazelaar, J. M., "Database Trends", Byte V14, N9, Setembro 1989.

[Tecn88] "Arco-Iris - Sistema Interativo de Acesso a Banco de Dados - Manual do Usuário", Versão 1.0, TecnoSoft Tecnologia de Software Ltda., Janeiro 1988.

[Tecn89a] "Arco-Iris - Módulo Gráfico", Versão 1.0, TecnoSoft Tecnologia de Software Ltda., 1989.

[Tecn89b] "A Interface de Programação do Arco-Iris para o Turbo Pascal", Versão 1.0, TecnoSoft Tecnologia de Software Ltda., 1989.

[Trot89] Trotta, Claudio Newton Ferreira, "Extensões no SGBD Copperel para Aplicações Não Convencionais", Tese de Mestrado em Engenharia de Sistemas e Computação, COPPE/Sistemas, U.F.R.J., Rio de Janeiro, Março 1989.

[Turb87] Turbyfill, C., "Comparative Benchmarking of Relational Database Systems", Ph.D. Thesis No.87-

871, Department of Computer Science, Cornell University, Ithaca, New York, Setembro 1987.

[Turb89] Turbyfill, Carolyn & Orji, Cyril & Bitton Dina, "AS3AP - An ANSI Sequel Standard Scalable and Portable Benchmark for Relational Database Systems", DB Software Corporation, Palo Alto, Califórnia, U.S.A., Setembro 1989.

[Ursc88] Urschel, William, "Paradox/386", PC World V6, N7, Julho 1988.

[Vose89] Vose, G. M. & Weil, Dave, "Benchmark Apologia", Dr. Dobb's Journal, Canada, Fevereiro 1989.

[Webs86] Webster, B., "Benchmarking", Byte, Janeiro 1986.

[Weic84] Weicker, R. P., "Dhrystone: A Synthetic System Programming Benchmark", Edgar H. Sibley Panel Editor, Communications of the ACM V27, N10, Outubro 1984.

[Whyt85] Whyte, Christine, "A Sense of Balance", PC World V3, N12, Dezembro 1985.

[Wong88] Wong, William, "Structured Query Language and Its LAN Alternatives", Micro/Systems, Setembro 1988.

[Xeph89] "Benchmarks for IBM Users", Xephon Consultancy Report, Junho 1989.

[Yao887] Yao, S. B. R Hevner, A., R. R Young-Myers, H., "Arialysis of Database System Architectures Using

Benchmarks", IEEE Transactions on Software Engineering V. SE-13, N6, Junho 1987.

[YuCT79] Yu, C. T. & Lam, K. & Siu, M. K. & Ozsoyoglu, "Performance Analysis of Three Related Assigment Problems", Proceedings of the 1979 ACM Sigmod International Conference on the Management of Data, Boston, Massachusetts, Maio - Junho 1979.

[Zaki82] Zakimi, B. et all, "COPPEREL: Sistema de Gerência de Bases de Dados da COPPE baseado em Algebra Relacional", Anais do IX SEMISH, Julho 1982.

[Zaki85] Zakimi, B. & Mattoso, M. & Prazeres, M., "Manual do Usuário da Loperel (Linguagem de Operação do SGBD Copperel)", Relatório Técnico ES-62/85, COPPE/Sistemas, U.F.R.J., Rio de Janeiro, 1985.

[Zant87a] "Zim/Installation and Operations Guide", Edition 2.0, Zanthe Information Inc., Setembro 1987.

[Zant87b] "Basic Concepts: How to Develop Applications with Zim", Edition 2.0, Zanthe Information Inc., Setembro 1987.

[Zant87c] "Zim/Data Dictionary & Command Reference Manual", Edition 2.0, Zanthe Information Inc., Setembro 1987.

[Zant87d] "Zim Version 3.0 Information", Zanthe Information Inc., Outubro 1987.

[Zant87e] "ZIM/DA Version 2.0 Information", Zanthe Information Inc., Outubro 1987.

[Zant87f] "ZINSTALL/2.0 - Layered Software Production Installation", Zanthe Information Inc., Outubro 1987.

[Zant87g] "ZIM/DA 2.0-Converting ZIM/DA 1.0 to run with ZIM /DA 2.0", Zanthe Information Inc., Outubro 1987.

[Zant87h] "The ZIM Error Help Facility", Zanthe Information Inc., Outubro 1987.

[Zant87i] "Zim Version 3.0 - User-defined Collating Sequence", Zanthe Information Inc., Outubro 1987.

[Zant87j] "Zim Version 3.0 - ZIM 2.5x Problem Resolution", Zanthe Information Inc., Outubro 1987.

[Zant87k] "Zim Version 3.0 - Release Notes", Zanthe Information Inc., Outubro 1987.

[Zant89] "Zim - Venha para Este Novo Mundo", Prospecto de apresentação do produto, Zanthe Information Inc. & RCM Informatica Ltda., 1989.

[Zuss88] Zussman, J. U. & Gomez, G., "SQL*Forms Designer's Reference", Version 2.0, Oracle Corporation, Fevereiro 1988.

APÊNDICE A

SISTEMAS GERENCIADORES DE BANCO DE DADOS UTILIZADOS

A seguir, serão brevemente caracterizados SGBD's existentes no mercado de microcomputadores, **selecionados, analisados e utilizados** nesta tese. São eles: Copperel, dBase III/Plus, Dialog Plus/c, Oracle, Paradox, Rbase V, e Zim.

Serão apresentadas as principais características de cada SGBD, respeitando os parâmetros estáticos definidos no capítulo III, para descrever e analisar a funcionalidade dos mesmos. Em alguns casos, achou-se necessário descrever aspectos referentes a modalidade multiusuária do produto, de forma a melhor entender seu funcionamento e desempenho.

A fim de facilitar o acesso as informações sobre os SGBD's, a sequência de apresentação é estritamente alfabética, não havendo, assim, nenhuma outra conotação na sua precedência. Algumas características serão omitidas por não terem sido explicitamente referenciadas na literatura.

As informações descritas a seguir, foram extraídas quase que inteiramente das referências bibliográficas citadas no início de cada seção. Isto se deve ao fato destes trabalhos serem as únicas fontes de informação disponíveis para obtenção das características necessárias a identificação da funcionalidade (parâmetros estáticos) de cada SGBD.

A.1 COPPEREL

A seguir, será apresentada uma descrição do SGBD COPPEREL. Informações podem ser obtidas em [Blum88, Blum89a, Blum89b, Gonç90, Matt85a, Matt85b, Matt85c, Matt85d, Matt87, Pale85, Souz82, Souz88, Trot89, Zaki82, Zaki85].

é um SGBD desenvolvido na COPPE/UFRJ, baseado no modelo relacional clássico. Além das facilidades normalmente encontradas nos sistemas relacionais, adiciona outras como: definição de asserções de integridade e modelos externos específicos usando relações virtuais, estrutura física ideal para implementar o modelo interno do banco de dados, mecanismos para autorização de usuários, procedimentos catalogáveis e, facilidade de auditoria e reconstrução.

a. Estrutura física e organização do espaço de armazenamento

Os dados são representados através de entidades e relacionamentos, e armazenados sob a forma de relações (tabelas). As propriedades desses objetos são descritas através de atributos (colunas) e as ocorrências são representadas por tuplas (linhas). São consideradas entidades: relações de base de dados, relações temporárias, asserções de integridade, procedimentos e variáveis.

A realização física de uma base de dados é feita em um único arquivo, que é dividido logicamente de modo a conter as diferentes relações, os índices, o esquema, a área de trabalho e as listas que implementam as relações de ordem.

Cada registro do arquivo pode conter uma ou mais tuplas de uma ou mais relações da base de dados. As tuplas que compõem uma relação estão armazenadas de forma contígua, e tuplas e registros possuem tamanho fixo. Porém, uma tupla não pode ser maior que um registro físico desse arquivo.

Os índices são do tipo aleatório, usando tabelas de espalhamento com resolução de colisões por endereçamento aberto e encadeando as tuplas de mesmo valor para o atributo indexado. Utiliza-se algoritmos de classificação na execução de comandos que envolvam critérios de comparação diferentes da igualdade, para os quais os índices não contribuem eficientemente.

Cada registro possui um identificador que guarda o seu endereço de armazenamento no arquivo físico. O endereço é formado pelo número do registro no arquivo e pelo deslocamento dentro do registro.

A remoção de tuplas ocasiona reorganização do espaço físico de armazenamento. As tuplas que ocupam as últimas posições físicas da relação são relocadas para posições vazias. Deste modo, nunca existem espaços vazios entre tuplas.

Vistas são relações cuja existência depende de outras relações. Estas não contêm dados, apenas ponteiros para os dados reais.

Desenvolveu-se uma técnica própria de organização do espaço de armazenamento, por paginação, que controla a alocação e liberação de buffers baseada nas características de localidade e frequencialidade das referências às tuplas.

reduzindo o tempo de paginação. A área de buffers é, também, usada pelas tuplas das relações do esquema por possuírem tratamento igual as tuplas do usuário. A política de substituição de páginas é chamada NRU ("Not Recently Used"), sendo uma variação da política LRU ("Least Recently Used"), retirando da memória a menos recentemente usada e podendo-se influenciar na escolha da página a ser retirada. É feito um rodízio entre páginas de buffers; e a tabela de mapeamento funciona como uma tabela de envelhecimento que informa o grau de uso da página pela sua posição na tabela.

Não existem limites para o número e o tamanho dos objetos armazenados. Estes são determinados somente pelo tamanho da memória secundária disponível.

A definição do modelo de referência da base de dados é feita por comandos LOPEREL. Há três níveis de representação de dados: nível conceitual, nível interno e nível externo.

A definição do esquema conceitual é feita pela descrição das entidades, atributos e relacionamentos que compõem a base de dados. As entidades e relacionamentos são tratados como relações. Há um esquema conceitual para cada base de dados gerenciada pelo COPPEREL, e as relações destes esquemas são geradas, automaticamente, durante a criação destas bases, juntamente com suas próprias descrições. Usuários autorizados podem consultar as relações do esquema, mas atualizações são feitas por comandos COPPEREL.

A definição do esquema interno é feita pela descrição das estruturas de armazenamento e estratégias de acesso. São definidas chaves de acesso, identificadores de entidades,

atributos indexados, tamanho e tipo dos atributos. Em alguns casos, ocorre a necessidade de alterar e reorganizar as estruturas de armazenamento para melhorar o desempenho do sistema; sendo possível criar e suprimir índices sobre atributos. A representação das relações é feita em um único arquivo físico, conforme descrito anteriormente.

A definição do esquema externo é feita pela descrição das entidades e relacionamentos do esquema externo, e do mapeamento do esquema conceitual e externo. As relações que o compõe podem ser vistas ou relações propriamente ditas.

b. Tipos de dados

Os tipos de dados suportados pelo COPPEREL são:

i. Caracter

Cadeia de caracteres com tamanho máximo de 256 bytes. Cada caracter ocupa 1(um) byte.

ii.Real

Valores numéricos com/sem sinal entre -1 X (10**38) e +1 X (10**38), com comprimento máximo de 4 bytes. A precisão fica dependente da instalação.

iii.Inteira

Valores inteiros com/sem sinal entre -214.748.364 e +214.748.364, com comprimento máximo de 4 bytes. A precisão fica dependente da instalação.

iv.Natural

Valores inteiros positivos entre 0 e 65.536, com comprimento máximo de 4 bytes. A precisão fica dependente da instalação.

v. Data

O tipo de dados Data foi definido na especificação do COPPEREL, entretanto, ainda não foi implementado.

c. Dicionário de dados

O SGBD não tem um dicionário de dados padrão para guardar informações sobre objetos de aplicações e estruturas do banco de dados. Funções do dicionário de dados são supridas por esquemas que armazenam informações e podem ser consultados por usuários via comandos LOPEREL.

Segundo está descrito no item n, a máquina virtual é composta de três sub-módulos: BSTRAP, SUPESQ e SNTTCO. Estes módulos são responsáveis pela geração dos esquemas.

O BSTRAP gera o arquivo "BOOTSTRAP" (esquema principal) que guarda informações sobre a representação dos dados a nível conceitual. As bases de dados são configuradas através deste arquivo, e são carregadas com as relações do esquema se autodescrevendo. Usuários autorizados podem alterar as tabelas do esquema usando comandos LOPEREL.

O SUPESQ gera o superesquema do COPPEREL que é composto de duas tabelas: TBD (TAB-BASE-DADOS) e TUB (TAB-USUARIO-BASE)

que guardam informações descritivas sobre as bases de dados gerenciadas pelo SGBD e, administram a criação, abertura e fechamento das mesmas [Matt85a]. Estas tabelas não pertencem a uma única base de dados e ficam armazenadas em arquivos distintos, diferenciando-se das demais relações do esquema. A TUB associa os usuários à bases de dados as quais estes estão autorizados a fazer acessos. A TBD guarda informações sobre as bases de dados cadastradas no SGBD.

d. Linguagem de definição e manipulação de dadas

Uma linguagem autecorrigida, denominada LOPEREL (Linguagem de Operação do COPPEREL), dispensa o uso de linguagem hospedeira e permite acesso abrangente e fácil ao banco de dados. Esta contém comandos para definição dinâmica, atualização, gerência e manipulação de bases de dados relacionais, controle de fluxo, expansão de procedimento, cálculo de expressão, além de comandos de E/S. As operações podem ser realizadas sobre conjuntos (com operadores da álgebra relacional) ou registro a registro (com comandos iterativos).

Os comandos e mensagens da linguagem são em português, que facilitam a comunicação com o usuário e seu aprendizado. Os comandos são em formato livre e a maioria das palavras reservadas possui abreviaturas equivalentes que podem ser usadas sem perda de compreensão e significado. As mensagens de erro foram implementadas de para auxiliar o usuário na construção de comandos sintática e semanticamente corretos.

Telas de auxílio estão disponíveis para permitir ao usuário obter informações sobre a sintaxe dos comandos. Deste modo, auxiliam na codificação de programas e depuração de erros.

Os comandos de consulta são escritos sempre através de relações operando e resultado. Porém, as relações que contêm o resultado podem ser criadas implicitamente pelo sistema. Assim, pode-se realizar consultas compostas, onde o resultado de uma consulta serve de entrada para outra.

A entrada de dados e comandos pode ser feita em massa através de blocos de registros lidos via arquivos externos ou registro a registro via teclado. Cabe ao usuário decidir a alternativa mais viável para a aplicação em questão. Por não possuir interface gráfica e telas formatadas de entrada de dados, a adição de registros via teclado deve obedecer o formato dos campos pré-definido na criação do arquivo, isto é, os registros devem ter todos os seus campos preenchidos na sequência correta e com formato preciso para evitar erros de entrada de dados.

A recuperação dos dados de saída pode ser feita em tela, em listagem de impressora ou em arquivo físico padrão externo à base de dados. Em ambos os casos, a emissão de dados pode ser feita usando-se o padrão da linguagem ou especificando-se caracteres de formatação, para emitir relatórios mais sofisticados. Assim, o usuário pode adaptar a apresentação dos dados de saída às suas necessidades. Não é possível gerar formulários e telas formatadas de saída de dados pela inexistência de uma interface gráfica.

é permitido ao usuário, também, catalogar procedimentos que definem comandos a serem executados quando da sua chamada. Estes procedimentos podem passar parâmetros que podem ser qualquer estrutura sintática da linguagem. Deste modo, o usuário final pode interagir com o SGBD mais facilmente.

Na versão multiusuária, proposta em [Pale85], a LOPEREL não sofreria alterações. O usuário interagiria com o sistema através de terminal remoto, como na versão monousuária.

e. Interface com usuário

Não existe, hoje, uma interface interativa por menus disponível que permita ao usuário selecionar as tarefas a serem realizadas por menus. É necessário conhecer a sintaxe dos comandos de forma a programar uma aplicação.

Não há interface gráfica para construir telas formatadas de entrada de dados para a adição de registros ao banco de dados. O usuário deve adicioná-los interativamente seguindo o formato dos campos pré-definidos na criação dos arquivos. Assim, não se pode gerar telas formatadas de saída e usá-las como um lay-out otimizado na geração de relatórios.

O COPPEREL não dispõe de interfaces que permitam acessar/fornecer dados e aplicações de/para outros SGBD's. Não há, também, como convertê-los para usá-los com sucesso.

O COPPEREL pode ser usado interativamente ou por lotes. Na linguagem LOPEREL estão presentes comandos que se prestam

às duas formas de uso, para dar o máximo de recursos que possibilitem ao usuário consultar e atualizar seus dados.

O sistema possui, também, interface para linguagens hospedeiras. A seguir, no ambiente operacional, será apresentada uma descrição mais detalhada desta interface.

f. Integração com outros SGBD's

Não há um mecanismo para migrar aplicações e dados de/para outros SGBD's. Adicionalmente, não há recursos que permitam acessar dados de outros produtos internamente ao COPPEREL.

g. Qualidade da documentação

A documentação disponível sobre o COPPEREL é boa, deixando a desejar, somente, nos manuais de operação do produto. O suporte é dado pelos projetistas que o conceberam. São conhecidos detalhes fundamentais de implementação que ajudam na sua adaptação à vários ambientes, e permitem sua extensão para incorporar novas características e potencialidades, melhorar seu desempenho e funcionalidade, e usá-lo em outros tipos de aplicações não convencionais. Os manuais, mensagens, comandos e telas são em português, facilitando a compreensão e aprendizado pelos usuários.

h. Segurança e proteção dos dados

A segurança e proteção aos dados é feita pela definição de senhas. Apenas usuários autorizados podem acessar e alterar as informações. A integridade da base de dados é garantida pela definição de procedimentos de validação dos dados e verificação das restrições especificadas. A verificação das autorizações é feita pelo COPPEREL mediante análise das definições armazenadas no superesquema (tabela TUB).

i. Deteção de Falhas e reconstrução do banco de dados

O Subsistema de Reconstrução do SGBD COPPEREL (SR) recupera dados em caso de falhas de transação, sistema e meio de armazenamento [Gonç90]. Usa-se o conceito de transação como unidade lógica de trabalho, no sentido de transformar uma base de dados de um estado consistente inicial para um novo estado consistente final, e o conceito de reconstrução, no sentido de executar procedimentos de recuperação de falhas que assegurem a consistência e integridade dos dados.

Transações com atualização em dados ativam o SR que copia a imagem dos dados num arquivo BI ("Before Image") antes das modificações. Falhas de transação levam o SR a desfazer seu efeito, recuperando do BI a imagem anterior dos dados modificados pela mesma e substituíndo-a na base de dados para voltar ao estado consistente do início da transação.

Transações de atualização executadas com sucesso, são registradas num arquivo AI ("After Image"). Em falhas de meio, deve-se restaurar o banco de dados com cópia ("backup") praviamente gerada e reexecutar as transações

armazenadas no AI, recriando o estado corrente dos dados no momento da falha. Para tanto, é necessário que o administrador do banco de dados gere cópia do mesmo antes de iniciar cada sessão ou que o usuário descarregue sua base de dados num arquivo "backup" sempre que achar ideal.

j. Concorrência

Há uma proposta de implementação multiusuária para o SGBD COPPEREL [Pale85]. Porém, esta não foi implementada. A seguir, serão descritas suas principais características.

Em ambientes multiusuário, SGBD, programas e banco de dados poderiam ser compartilhados. Seria necessário um mecanismo de bloqueio para garantir a segurança e integridade dos dados. Para tanto, utilizaria-se um módulo com filosofia de monitor chamado núcleo que centralizaria e monitoraria o atendimento dos pedidos de acesso à base de dados.

Seria utilizada a estrutura física da versão monousuária acrescida de características que permitiriam acesso concorrente à base de dados. Utilizaria-se uma única base de dados de uso comum, que residiria em um arquivo físico em disco contendo todos os arquivos (tabelas) dos usuários.

k. Ambiente Operacional

Em ambientes monousuário que usam microcomputadores do tipo PC-XT ou PC-AT, o SGBD necessita de uma configuração mínima

de hardware de 640Kb de memória RAM e 2 Mb em disco rígido, e sistema operacional MS-DOS na versão 3.1 ou posterior.

Na versão COPPEREL-PC instalada, o módulo executável ocupa 390K de memória com estrutura de "overlay". Sendo assim, o espaço restante de memória pode ser aproveitado para aumentar os buffers internos do SGBD, além de permitir as novas extensões previstas [Souz88].

1, Portabilidade e flexibilidade

Desenvolveu-se o COPPEREL usando técnicas de portabilidade, para permitir sua instalação em diferentes ambientes. Deste modo, independe das características do sistema operacional hospedeiro e das aplicações a serem desenvolvidas.

Existem versões do COPPEREL instaladas em microcomputadores do tipo PC-XT ou PC-AT, computadores de grande porte como o IBM/370, IBM-4341 e IBM-4381 com sistema operacional VM que possue características de uma máquina virtual e o UNISYS B6800, e estações de trabalho com sistema operacional UNIX.

m. Gerador de Aplicações

Não existe, hoje, um gerador de aplicações para permitir ao usuário desenvolver e alterar aplicações sem programação

n. Arquitetura

Sua arquitetura modular é composta de dois módulos principais: a máquina virtual e o compilador da Loperel.

A máquina virtual é responsável pela execução dos comandos fornecidos pelo usuário e é composta de três sub-módulos: BSTRAP, SUPESQ e SNTTCO. O BSTRAP é responsável pela inicialização do SGBD, com a geração do arquivo BOOTSTRAP usado para inicializar a base de dados de trabalho e guardar informações sobre a representação dos dados a nível conceitual (esquema), pelo cálculo dos endereços e pela alocação do espaço físico das tabelas do esquema e seus respectivos índices. O SUPESQ é responsável pela geração do superesquema do COPPEREL e pela administração do banco de dados, e será descrito, a seguir, no "item o". O SNTTCO é responsável pelos acessos à base de dados de trabalho.

O compilador é responsável por receber comandos LOPEREL e realizar análises léxica e sintática, e testes semânticos com chamadas à máquina virtual. A tabela de símbolos é formada por relações do sistema, e a geração de instruções pelo compilador é feita em código intermediário para ser executado na máquina virtual. Pode-se adicionar novos comandos à linguagem via i : " : na gramática e definição de código intermediário, executável em novos módulos que devem ser incorporados à máquina virtual.

o. Utilitários

Para gerenciar as bases de dados cadastradas no SGBD, existe um utilitário chamado SUPESQ que controla a criação,

abertura e fechamento das mesmas, recupera-as em caso de falhas de hardware e define autorizações. A administração de cada base de dados, individualmente, é feita usando-se recursos da linguagem LOPEREL pois cada base de dados se constitui numa materialização do esquema e dos dados.

p. Outros (Extensão do SGBD à aplicações não convencionais)

O COPPEREL-PC vem sendo explorado para ser extendido, poder ser usado a classes de aplicações específicas e beneficiar vários projetos [Souz88]. A seguir, serão apresentados de forma sucinta, alguns destes projetos. Informações adicionais podem ser obtidas na bibliografia referenciada.

Em [Blum88, Blum89a, Blum89b], é apresentado um protótipo de Sistema de Gerência de Banco de Dados Orientado a Objetos (CPRELOBJ), implementado a partir da extensão do COPPEREL. Neste protótipo, o COPPEREL é usado no nível de gerência dos dados armazenados, e a extensão consiste no desenvolvimento de um outro nível de sistema, orientado a objetos, que use o COPPEREL como base e forneça ao usuário características de um ambiente de banco de dados orientado a objetos. O CPRELOBJ suporta a definição e manipulação de objetos compostos, organizados semanticamente segundo uma hierarquia, com herança simples de propriedade.

Em [Trot89], o COPPEREL foi extendido de forma a possuir interface para linguagens hospedeiras e suportar objetos complexos e campos longos. A construção da interface visou permitir o uso do COPPEREL via linguagens de programação de

uso geral, próprias para a construção de determinadas aplicações. A inclusão de campos longos permitiu a construção de um gerente geral de armazenamento de objetos e a representação de um novo tipo de dados constituído de uma sequência de bytes arbitrariamente longa e não interpretada. A incorporação de objetos complexos conservou a arquitetura geral do SGBD, permitiu tratar o enfoque estrutural da orientação a objetos e forneceu alguma flexibilidade e operações de alto nível ao SGBD.

O objetivcr das extensões é adaptar o COPPEREL às aplicações não cosivesicionais que enVolVem o processamesi9o complexo de grande volume de dados, para permitir desenvolver ap1ieações sia área dc ambien9es de software, au1omação de escritórios, inteligência artificial, CAD/CAM e bancos de dados textuais a pi:9óri:os. Porém, o 8GBD sião pei-deu suas características relacionais, podendo o usuário dispor dea2as fac ilidades dur-ante a implemen1ação de ap1icações.

A.2 DBASE III/PLUS

A seguir, será apresentada uma descrição do DBASEIII/PLUS. Informações podem ser obtidas em [Bara88, Barn88, Cose86a, Cose86b, Curo86, Data85a, Data85b, Data86a, Data86b, Data86c, Data86d, Data86e, Data89a, Data89b, Data89c, Data89d, Derf88, Dick86a, Edel89a, Eise86, Elet87, Hart85, Hart86, Hart87a, Hart87c, Info88d, Info89, Inte…, Juni86, Lent89, Luzi87, Nogu86, Olym88, Perr88, Petr87, Poor87, Rama88b, Rama88c, Rama88d, Rama88e, Rube87, Rube89,

Sant87a, Seym88c, Sham88, Shaw90a, Shaw90b, Smi187, Star90].

é um SGBD relacional desenvolvido pela Ashton-Tate e, representado e comercializado no Brasil pela Datalógica.

a. Estrutura física e organização do espaço de armazenamento

Os dados são armazenados em tabelas de duas dimensões organizadas em linhas (registros) e colunas (campos). Os registros possuem tamanho fixo. Relacionamentos entre arquivos são estabelecidos através de seus campos e são do tipo um-para-um ou muitos-para-um, pois o DBASE III/PLUS não tem capacidade para recuperar vários registros em relacionamentos um-para-muitos.

O número de tabelas, em uso concorrentemente, é limitado a 10, enquanto o número de arquivos de índice por base de dados ativa é limitado a 7. Cada tabela pode ter até 1 (um) bilhão de registros com até 128 campos (4000 bytes) por registro. Cada campo pode ter até 255 caracteres.

Deleções em registros não liberam, automaticamente, o espaço em disco. Cabe ao administrador do banco de dados reorganizar o espaço, para liberar áreas não mais usadas.

Buffers são utilizados para armazenar os últimos comandos diretos digitados no modo interativo, que foram executados pelo SGBD. O usuário pode definir o número de comandos a serem guardados, historicamente, no buffer.

Variáveis de memória criadas durante uma sessão DBASE III/PLUS, podem ser armazenadas em um arquivo de memória de forma a serem preservadas ao fim da sessão ou programa que as criou. A quantidade de variáveis de memória ativas fica limitada a 256, com tamanho total de 6.000 bytes.

Usuários podem criar arquivos especiais contendo especificações para a formatação de relatórios, telas e etiquetas, condições de seleção para a restrição de acesso a registros do banco de dados e informações resultantes da execução de comandos dBase.

Os registros são acessados/armazenados fazendo uso do método sequencial indexado, implementado usando árvore B+ como estrutura de armazenamento. O usuário pode ordenar os registros de uma base de dados por uma determinada chave. O DBASE III/PLUS monta um arquivo de índices usando árvore B+ associado a esta base de dados, para atender a sequência lógica dos dados requisitados pelo usuário e otimizar a busca. Atualizações nos dados mantêm os registros ordenados e não prejudicam a busca de informações.

b. Tipos de dados

Os tipos de dados suportados pelo DBASE III PLUS são:

i. C (Caracter)

Cadeia de caracteres com tamanho máximo de 254 bytes. Cada caracter ocupa 1(um) byte.

ii.N (Numeric)

Valores numéricos com/sem sinal com comprimento máximo de 19 bytes. A precisão fica limitada aos 15 dígitos mais significativos.

iii.L(Logic)

Valores lógicos (True/False, Y/N) ocupando 1(um) byte de espaço de armazenamento.

iv.M(Memo)

Cadeia de caracteres em formato texto, armazenada em arquivo auxiliar de campos memo, utilizando espaço de armazenamento de 10 bytes em cada registro do arquivo de banco de dados. Cada caracter ocupa 1(um) byte e o texto é dividido em blocos de 512 bytes até uma extensão máxima de 5.000 bytes.

v. D(Date)

Datas no formato especificado pelo usuário, ocupando 8 bytes de espaço de armazenamento.

c. Dicionário de dados

Arquivos são organizados em catálogos e visões. Catálogos guardam os arquivos a serem usados enquanto visões guardam subconjuntos do catálogo e podem especificar níveis de acesso à campos do banco de dados. Porém, cabe ao usuário gerenciar estes dois subconjuntos que funcionam como um dicionário de dades não automatizado de uma base de dados.

d. Linguagem de definição e manipulação de dadas

Uma linguagem de comandos interativa e interpretada (dBASE) permite ao usuário desenvolver programas, inspecionar suas execuções via comandos depuradores, manipular e criar arquivos de dados, e definir telas de E/S de dados.

Pode-se usar o editor de textos do DBASE ou qualquer outro editor para criar programas. Comandos não possuem posição fixa na linha de comando, podendo delimitá-los por vários brancos (inclusive antes do primeiro comando na linha). As palavras reservadas podem ser abreviadas pelos 4 primeiros caracteres, e não devem ser usadas para qualificar nomes de arquivos, campos e variáveis de memória.

Telas de auxílio permitem ao usuário obter informações sobre operações e sintaxe de comandos. Estas telas auxiliam na codificação de programas e depuração de erros, e possuem um índice de auxílio por tópico de programação.

A entrada de dados e comandos pode ser feita em massa via leitura de arquivos externos ou via tela que deve conter os campos do arquivo a serem atualizados. Telas podem conter informações de vários arquivos e definições de regras de validação para a conferência dos valores fornecidos pelo usuário quanto a formato, comprimento e conteúdo.

As regras não são associadas aos campos no arquivo físico e ficam embutidas na tela na qual foram criadas. Assim, a entrada de dados em arquivos que forem feitas sem usá-las, não serão criticadas. As regras sofrem outras limitações como a não permissão da inclusão de comandos dBase na sua

especificação, a não inclusão destas no dicionário de dados e, a especificação destas no comando que trata a entrada de dados a cada tela. Procedimentos de validação adicionais devem ser embutidos nos programas que ativam as telas, para validar os dados antes de atualizar o arquivo.

A linguagem dBase dispõe, também, de comandos para emissão de relatórios. A especificação de caracteres de formatação pelo usuário, adapta o relatório ao formato desejado.

e. Interface com usuário

Uma interface interativa por menus ("Assistant") permite ao usuário consultar e manipular o banco de dados via seleção de comandos e, definir telas, relatórios, formulários e outros objetos associados com os arquivos.

As telas estão divididas em 3 áreas. A primeira é a área de menu, que mostra opções de comando. A segunda é a área das janelas, que contém menus internos, submenus, formulários, tabelas e outros objetos de trabalho. A terceira é a área de mensagens onde são apresentadas informações sobre a sintaxe do comando corrente (linha de ação), ambiente, operação e objeto atuais (barra de estado), instruções para movimentação na tela e seleção de uma opção, e mensagens de erro (linha de navegação) e, operações e entradas a serem realizadas no menu ativo (linha de mensagem).

Interfaces permitem acessar/fornecer dados de/para arquivos tipo WKS (Lotus 123), DBF (dBase II e III), DIF (VisiCalc), PFS (IBM Professional Files), SYL (Multiplan) e ASCII.

Outra interface permite chamadas de rotinas em C e Assembler internamente aos programas.

Porém, o SGBD não possui uma interface de alto nível que permita a visualização do banco de dados como uma coleção de tabelas. O usuário precisa conhecer a estrutura interna do banco de dados e gerenciar arquivos de índices, dados e relacionamentos, catálogos e visões.

f. Integração com outros SGBD's

A importação/exportação de dados entre bancos de dados é feita no formato ASCII, VisiCalc, Multiplan, Lotus 123 e PFS. Entretanto, os dados devem possuir formato e comprimento iguais no arquivo fonte e destino. Não há integração entre o DBASE III PLUS e outros SGBD's.

g. Qualidade da documentação

A documentação sobre o DBASE III PLUS é vasta. O suporte ao SGBD é dado pelos representantes nacionais. Seus manuais são em português, embora seus comandos, mensagens e telas sejam em inglês como os demais produtos estrangeiros, dificultando a compreensão de usuários leigos neste idioma.

h. Segurança e proteção dos dados

A proteção dos dados é feita mediante definição de senhas e níveis de acesso ao banco de dados. Os dados podem ser acessados somente por usuários autorizados.

O esquema de proteção é feito pelo utilitário "Protect" que controla os tipos de acesso autorizados a cada usuário e determina a forma como os dados devem ser armazenados. É possível estabelecer proteção a nível de usuário, arquivo, registro e campo. Adicionalmente, pode-se produzir uma versão criptografada do arquivo, sem perder a versão original a qual não deve ser deixada disponível para garantir proteção integral aos dados.

i. Deteção de Falhas e reconstrução do banco de dados

Não há um mecanismo restaurador automático pava vecuperar os dados, anular as transações incompletas e garantir a segurança e integvidade dos irifovmações em caso de falhas de software/hardware ou falta de energia. Não é utilizado um niecarisaio automatizado para garar cópiз dc bancr de dados antes de cada sessão. Deste modo, os arquivos podem ser danificados e perderem parte ou todas as informações. Cabe ao usuário gerar e manter cópias do banco de dados, de forma a permitir a recuperação de dados.

j. Concorrência

Em ambientes multiusuário (em redes), arquivos e registros podem ser compartilhados. Um mecanismo de bloqueio garante

proteção, segurança e integridade aos dados durante atualizações no banco de dados. Este mecanismo assegura que um arquivo pode ser lido por vários usuários ao mesmo tempo, embora só possa ser atualizado por um único usuário.

Há vários tipos de bloqueio. São eles:

i. Modo exclusivo

Somente um usuário pode acessar o arquivo ao mesmo tempo.

ii. Modo compartilhado

Vários usuários podem acessar o arquivo ao mesmo tempo. Porém, um mecanismo de bloqueio automático bloqueia o arquivo em uso durante operações de atualização e libera-o ao final da transação.

iii. Bloqueio por registro

Bloqueia o registro para evitar que diferentes usuários usem-o ao mesmo tempo. Permite a execução de transações em série.

Para evitar a ocorrência de "deadlocks", há funções de proteção para verificar se arquivos e registros estão bloqueados por algum usuário, antes de usá-los. Arquivos e registros desbloqueados pela função 'bloqueados pelo usuário executor da função, impedindo o acesso de outros usuários aos mesmos até o fim da transação.

k. Ambiente Operacional

Em ambientes monousuário que usem microcomputadores do tipo PC-XT ou PC-AT, é necessária uma configuração mínima de hardware de 384Kb de memória RAM, 2.5Mb em disco rígido e sistema operacional MS-DOS versão 2.1 ou posterior. Porém, em ambientes multiusuário, é necessário 640 KB de memória RAM e sistema operacional MS-DOS versão 3.1 ou posterior.

1. Portabilidade e flexibilidade

O DBASE III/PLUS não foi desenvolvido usando técnicas de portabilidade, não podendo ser instalado em qualquer ambiente. Assim, depende das características do sistema operacional hospedeiro.

Há versões instaladas em microcomputadores do tipo PC-XT ou PC-AT. Porém, não há versões disponíveis para computadores de grande porte, apesar de haver uma versão para ambientes Macintosh [Sham88].

m. Gerador de Aplicações

O Gerador de Aplicativos (GA) permite definir e criar bases de dados, telas, relatórios, menus e tarefas necessárias a uma aplicação, sem programação. Sua interface interativa por menus dispensa o usuário do conhecimento da sintaxe dos comandos. As telas geradas pelo GA podem ser, posteriormente, modificadas pelo GA ou pelo usuário, e possuem comandos no formato padrão da linguagem dBase.

Não pode-se gerar aplicativos pelo GA sem o uso de menus. Logo, torna-se inviável sua aplicabilidade a esta tese que fará a entrada de dados e comandos das aplicações em massa via leitura de arquivos externos sem o uso de telas, para eliminar a improdutividade gasta com a seleção de opções em menus e, digitação e fornecimento de informações via tela.

O Gerador de Relatórios permite construir relatórios interativamente, podendo acessar vários arquivos. O formato dos relatórios pode ser adaptado às necessidades da aplicação, através de caracteres de formatação. Apesar de não ser possível gerar relatórios sofisticados e flexíveis, existem geradores externos ao DBASE III/PLUS que agem sobre arquivos de dados gerados pelo SGBD e produzem relatórios mais elaborados.

O Gerador de Telas permite criar telas sem programação e com a seleção de comandos por menus. As telas podem conter informações de vários arquivos.

n. Arquitetura

Não foi possível obter informações sobre a arquitetura do SGBD. A documentação disponível não relata detalhes de implementação e da arquitetura usada pelo DBASE III/PLUS.

o. Utilitários

Um Sistema de Consulta Avançado permite formular, obter e guardar os resultados de consultas a partir da seleção de

comandos em menus. Os resultados das consultas podem ser armazenados em arquivos para permitir o acesso às informações por outros programas e aplicações.

O "RunTime" gera módulos executáveis criptografados. O código da aplicação torna-se ilegível e protegido contra cópias, linhas de comentários e tabulações são eliminadas para reduzir o tamanho dos programas e executá-los mais rapidamente, e programas separados são grupados em um único programa aplicativo executável somente pelo dBRUN e DBASE III/PLUS. Assim, aplicações são executadas sem a presença do código fonte. O dBRUN é um interpretador do DBASE III/PLUS, projetado para ler arquivos encriptografados e que dispensa a presença do SGBD na sua execução.

Em ambientes multiusuário (em redes), o utilitário "dBase Administrator" permite administrar a utilização do SGBD na instalação em uso, gerenciando o compartilhamento dos dados e programas, o uso concorrente dos mesmos e o acesso aos dados por usuários autorizados. Assim, o mecanismo de segurança ("Protect") e o mecanismo de bloqueio são controlados pelo "dBase Administrator".

Segundo [Cust 86], o "dBase Administrator" dificulta a instalação do hardware a software adequados ao ambiente, idiciorindo estições da trabalho ("workstations"). Estas dificuldades sao atribuídas à forma de proteção e ao mecarismo de fechamento de registroe e arquivos iriteiros em determinados comandos. Não foi possível analisar tais rificuldides no imbierite tperAtionAl em uso nesta tese, pelo fato deste ser monousuário.

Para permitir acesso aos arquivos e, execução de aplicações e programas desenvolvidos em DBASE III/PLUS em empresas que não o usam, há no mercado compiladores da linguagem dBASE [Hart86, Hart87a, Rube87]. Geralmente, não é permitido ao usuário final alterar uma base de dados sem o controle do SGBD. Logo, muitos comandos não são aceitos pelos compiladores para manter a integridade do banco de dados.

Com o advento dos compiladores dBASE, os desenvolvedores de software podem entregar aplicações ao usuário final na sua versão executável. Os módulos fonte não são entregues, preservando sua integridade. Alterações na aplicação podem, somente, ser feitas pelos desenvolvedores. Em empresas que não possuem o SGBD, o custo para adquirir um compilador é inferior ao custo de compra do produto, e a velocidade de execução da aplicação é superior nos compiladores.

p. Outros (Extensão do DBASE III/PLUS)

Apesar do DBASE III/PLUS não usar a linguagem relacional SQL para definir a manipular dados, o DBASE IV faz uso do padrão IBM. Contudo, o modo de programação é diferente do descrito anteriormente, inibe alguns comandos dBASE que facilitam os seus comandos e torna lenta consultas à tabelas por comandos SQL, devido a passos intermediários necessários para tradução destes em comandos dBASE [Rube89]. Pelos motivos especificados no capítulo IV, não foi possível utilizar o DBASE IV nesta tese.

Outros produtos estão disponíveis no mercado para interagir com o DBASE III/PLUS e aumentar sua funcionalidade [Hart85, Hart87c]. Estes incluem geradores de relatórios, gráficos, programas e telas, depuradores e indexadores de arquivo. Estas ferramentas podem ser executadas em instalações que não utilizem o SGBD, pois atuam sobre arquivos de dados.

A.3 DIALOG PLUS/C

A seguir, será apresentada uma descrição do DIALOG PLUS/C. Informações podem ser obtidas em [Curo86, Info88b, Info88f, Magr88, Soft88a, Soft88b, Soft89a, Soft89b, Soua88].

é um SGBD relacional desenvolvido, representado e comercializado no Brasil pela Soft Consultoria em Processamento de Dados Ltda.

a. Estrutura física e organização do espaço de armazenamento

Os dados são armazenados em tabelas de duas dimensões, organizadas em linhas (registros) e colunas (campos). é possível estabelecer até 9 relacionamentos do tipo um-para-um e um-para-muitos entre tabelas. Os relacionamentos são estabelecidos através de campos de arquivos.

Cada arquivo pode ter até 1 bilhão de registros com 2000 campos cada (65535 bytes por registro e 512 bytes por campo). Uma aplicação pode acessar até 10 arquivos de banco de dados ao mesmo tempo e dispor de até 7 arquivos de

índice para cada banco de dados ativo. O conteúdo da chave de indexação é limitado a 255 caracteres.

Deleções em registros não liberam, automaticamente, o espaço em disco. Cabe ao administrador do banco de dados reorganizar o espaço, para liberar áreas não mais usadas.

Buffers são usados para armazenar os últimos comandos diretos digitados no modo interativo, que foram executados pelo SGBD. O usuário pode definir o número de comandos a serem guardados, historicamente, no buffer. Estes podem ser recuperados, reeditados e resubmetidos ao SGBD.

Variáveis de memória criadas durante uma sessão DIALOG, podem ser armazenadas em um arquivo de memória para serem preservadas ao fim da sessão ou programa que as criou. A quantidade de variáveis de memória ativas fica limitada pelo tamanho da memória principal disponível, enquanto cada linha de comando pode conter até 255 caracteres.

Usuários podem criar arquivos especiais contendo condições de seleção para restringir acessos a registros do banco de dados, especificações para a geração de relatórios, telas e etiquetas e informações de execuções de comandos Dialog.

Os dados são acessados/armazenados fazendo uso do método sequencial indexado implementado utilizando árvore B+ como estrutura de armazenamento. O usuário pode c : : os registros de uma base de dados pela ordem desejada e fazer acesso direto a um registro específico desta, através de arquivos de índice. A chave de indexação deve ser composta pelos campos que serão utilizados para otimizar a busca.

Arquivos de índice estão organizados em forma de árvore B+ com tamanho de nó fixo de 512 bytes. Ponteiros são lógicos, guardando números de registros lógicos. Não é considerado o deslocamento real de onde se pode encontrar os registros apontados. Ponteiros para níveis inferiores da árvore possuem tratamento igual àqueles para registros de dados.

b. Tipos de dados

Os tipos de dados suportados pelo DIALOG PLUS/C são:

i. C (Caracter)

Cadeia de caracteres utilizando um espaço mínimo de armazenamento de 8 bytes com alocação progressiva de 8 em 8 bytes até um tamanho máximo de 512 bytes. Cada caracter ocupa 1(um) byte.

ii.I (Numérico inteiro)

Valores inteiros com/sem sinal de até 9 dígitos, ocupando espaço de armazenamento de 8(oito) bytes.

iii.N (Numérico decimal)

Valores numéricos com/sem sinal de até 19 dígitos, ocupando espaço de armazenamento de 8(oito) bytes.

iv. L(Lógica)

Valores lógicos (Verdadeiro/Falso, S/N) ocupando 1(um) byte de espaço de armazenamento.

v. M(Memo)

Cadeia de caracteres em formato texto, armazenada em arquivo auxiliar de campos memo, utilizando espaço de armazenamento de 8 bytes em cada registro do arquivo de banco de dados. Cada caracter ocupa 1(um) byte e o texto é dividido em campos de 8 bytes.

vi.D(Data)

Datas no formato especificado pelo usuário, ocupando 4 bytes de espaço de armazenamento.

vii.Tabela

Vetor com até 65.535 elementos do mesmo tipo e tamanho. São aceitos todos os tipos anteriores, exceto o tipo M(memo). O espaço de armazenamento ocupado corresponde ao número de elementos da tabela multiplicado pelo tamanho de cada elemento.

c. Dicionário de dados

Arquivos são organizados em catálogos e visões. Catálogos guardam os arquivos a serem usados, independentemente do diretório/subdiretório a que pertençam e possuem estrutura idêntica à de um arquivo de banco de dados. Visões guardam subconjuntos do catálogo e podem especificar níveis de acesso à campos do banco de dados e estabelecer relações entre arquivos.

É possível realizar pesquisas no catálogo ativo no momento da busca e manipulá-lo pelos mesmos comandos de manipulação de base de dados. Em sessões que possuem um catálogo ativo,

somente os arquivos registrados neste, podem ser acessados. O catálogo funciona como um dicionário de dados não automatizado e restrito para guardar arquivos, relatórios, telas, visões, índices e consultas de uma aplicação, para grupar seus objetos e restringir o acesso a base de dados.

d, Linguagem de defiriição e manipulação de dados

Uma linguagem interativa e interpretada (Dialog) permite desenvolver programas para consultar e atualizar o banco de dados e criar aplicações. Os comandos em português e, a padronização da acentuação e caracteres decimais ao formato brasileiro simplifica o uso e a codificação de programas.

Os comandos não possuem posição fixa na linha de comando, podendo ser delimitados por brancos (inclusive antes do primeiro comando na linha). Palavras reservadas podem ser abreviadas até o menor número de caracteres que permita distinguí-las, e não devem ser usadas para qualificar nomes de arquivos, campos e variáveis de memória.

A entrada de dados e comandos pode ser feita em massa via leitura de arquivos externos ou via tela, que deve conter os campos do arquivo a serem atualizados. Cada arquivo pode estar associado à várias telas e uma tela pode realizar a entrada de dados em vários arquivos, simultaneamente. Campos podem possuir regras de validação para verificar os valores quanto a formato, comprimento e conteúdo.

As regras não são associadas aos campos no arquivo *físico* e ficam embutidas na tela ra qual foram criadas. Assim, a

entrada de dados em arquivos que forem feitas sem usá-las, não serão criticadas. As regras sofrem outras limitações como a não permissão da inclusão de comandos dialog na sua especificação, a não inclusão destas no dicionário de dados e, a especificação destas no comando que trata a entrada de dados a cada tela. Procedimentos de validação adicionais devem ser embutidos nos programas que ativam as telas, para validar os dados antes de atualizar o arquivo.

Telas de auxílio fornecem informações sobre operações, sintaxe de comandos, movimentações de telas e mensagens de erro, possuem índices por tópico de programação e auxiliam na codificação de programas e depuração de erros.

A linguagem Dialog dispõe, também, de comandos para a emissão de relatórios. A especificação de caracteres de formatação, adapta o relatório ao formato desejado.

e. Interface com usuário

Uma interface interativa por menus em português dispensa o conhecimento da sintaxe dos comandos, facilitando o uso e o aprendizado do SGBD. O usuário pode consultar e atualizar o banco de dados, e criar telas, relatórios, formulários e outros objetos DIALOG PLUS/C associados com arquivos.

Janelas de diálogo são utilizadas pelo SGBD para obter informações do usuário sobre parâmetros de comandos ou autorizações para execução de processos.

Interfaces permitem acessar/fornecer dados de/para arquivos tipo WKS (Lotus 123), DBF (dBase II e III), DIF (VisiCalc), PFS (IBM Professional Files), SYL (Multiplan), ASCII e Carta-Certa. Outra interface permite que aplicações em linguagens de terceira geração interajam com o SGBD.

f. Integração com outros SGBD's

A importação/exportação de dados entre bancos de dados é feita no formato ASCII, VisiCalc, Multiplan, Lotus 123, PFS e Carta-certa. Entretanto, os dados devem possuir formato e comprimento iguais no arquivo fonte e destino.

Há um conversor ("Importe/Exporte") que transforma arquivos DIALOG em dBASE e vice-versa, permitindo aproveitar, integralmente, os dados e programas. Entretanto, não há integração entre o DIALOG PLUS/C e outros SGBD's.

g. Qualidade da documentação

A documentação sobre o DIALOG PLUS/C é pouca. O suporte ao SGBD é dado pelo fabricante e representantes nacionais. Seus manuais, comandos, mensagens e telas são em português, facilitando a compreensão de usuários.

h. Segurança e proteção dos dados

A segurança dos dados é garantida pelo utilitário PROTEGE que permite a definição de senhas e níveis de acesso ao

banco de dados. O usuário só pode executar as operações e acessar os dados a ele autorizados. A proteção pode ser a nível de usuário, arquivo, registro e campo. O mecanismo de segurança permite, também, criptografar os dados para protegê-los contra leituras e atualizações indevidas.

i. Deteção de Falhas e reconstrução do banco de dados

Não há um mecanismo restaurador automático para recuperar dados, anular transações incompletas e garantir a segurança e integridade das informações em caso de falhas de software /hardware ou falta de energia. Não é usado um mecanismo automatizado para gerar cópia do banco de dados antes de cada sessão. Logo, os arquivos podem ser danificados e perderem parte ou todas as informações. Cabe ao usuário gerar e manter cópias do banco de dados, de forma a permitir a recuperação de dados.

j. Concorrência

Pode-se compartilhar informações com níveis diferenciados de acesso. A integridade dos dados é assegurada por um mecanismo de reserva que garante a proteção e segurança do SGBD. Registros de arquivos compartilhados ficam bloqueados durante alterações nos mesmos por algum usuário autorizado. O DIALOG PLUS/C reserva o arquivo automaticamente antes de serem executados comandos que possam alterá-lo. O arquivo é liberado no fim da transação.

k. Ambiente Operacional

Em ambientes monousuário que usem microcomputadores do tipo PC-XT ou PC-AT, o SGBD necessita de uma configuração mínima de hardware de 512Kb de memória RAM, 2,5 Mb em disco rígido, e sistema operacional MS-DOS, SISNE, SIM/DOS ou PC/DOS versão 2.1 ou posterior. Entretanto, em ambientes multiusuário, é necessário 640 KB de memória RAM e sistema operacional MS-DOS versão 3.1 ou posterior.

l. Portabilidade e flexibilidade

Há versões do DIALOG PLUS/C instaladas em microcomputadores do tipo PC-XT ou PC-AT, e ambientes SOD/C-500, UNIX-LIKE, XENIX, EDIX, SOX, SIDIX (SID) e UNIX_V/386. As aplicações desenvolvidas em Dialog podem ser migradas entre os ambientes especificados, com um mínimo de alterações.

A migração de aplicações desenvolvidas em microcomputadores do tipo PC-XT ou PC-AT para o ambiente UNIX-LIKE é feita pelo utilitário CONVIX, que converte os arquivos de banco de dados para um formato compatível com o ambiente. Os programas não sofrem alterações.

O GERANDX permite a criação de arquivos de índice em ambiente UNIX-LIKE. Estes arquivos necessitam de menor quantidade de espaço em disco para serem armazenados.

m. Gerador de Aplicações

A geração de aplicações é feita pelo Dialog-Ger, uma ferramenta que permite gerar programas para criação, reorganização, indexação, inclusão, exclusão, atualização e consulta a dados do banco de dados, e gerar bases de dados, telas e relatórios, sem programação. Deste modo, é possível gerar estruturas do banco de dados e programas de uma aplicação a partir do conhecimento do usuário.

As informações do usuário vão sendo armazenadas numa base de conhecimento e, ele pode, a qualquer momento, gerar uma versão preliminar da aplicação, mesmo se as informações estiverem incompletas. Modificações e inclusões são feitas nos dados e uma nova versão é gerada automaticamente.

A geração de gráficos é feita pelo Dialog-Graf que permite criar gráficos do tipo linha, torta e histograma-3D a partir de dados armazenados no banco de dados. Junto aos arquivos do SGBD, é instalada uma biblioteca gráfica que permite a geração dos gráficos através de comandos da linguagem. O usuário pode executar seus gráficos dentro do ambiente DIALOG PLUS/C em modo interpretado e compilado.

O SGBD dispõe de uma planilha eletrônica (Dialog-Fix) que permite a leitura de informações de bancos de dados, a geração de planilhas e a criação de arquivos com os resultados de cálculos feitos nas planilhas. Aquelas desenvolvidas em Lotus 123 podem ser migradas para o Dialog-Ger, através da tradução destas para o formato FIX.

O Dialog-Ger permite a transmissão automática de valores entre planilhas e a geração de relatórios e gráficos a

partir dos cálculos obtidos pela planilha. O tamanho desta independe da quantidade de memória disponível.

o. Utilitárias

Um compilador (CDIALOG) traduz arquivos de programas em módulos objetos em linguagem de máquina, de forma a acelerar a execução de aplicações. Porém, o compilador não dispõe de todos os comandos e funções do modo interativo interpretado, restringindo o escopo das operações que podem ser implementadas numa aplicação.

Os módulos objetos são gerados no padrão Intel 8086, e podem ser link-editados por qualquer link-editor que obedeça a este padrão. Em caso de erro, é gerada uma listagem dos erros ocorridos no programa compilado. Adicionalmente, pode ser gerada uma listagem de referências cruzadas para nomes de variáveis de arquivos e rotinas.

A Biblioteca de Acesso a Banco de Dados Dialog (BABDD) é formada por uma coleção de subrotinas agregadas em uma biblioteca de programas-objeto relocáveis. O método de acesso usado no gerenciamento de base de dados encontra-se de forma modularizada nesta biblioteca, que permite, também, o uso do DIALOG por programas em linguagens de terceira geração. Deste modo, aplicações desenvolvidas em outra linguagem podem interagir com o produto.

A.4 ORACLE

A seguir, será apresentada uma descrição do ORACLE. Informações podem ser obtidas em [Amad87, Brow87, Camp88, Cole87, Comp89a, Comp89b, Comp89c, Comp89d, Comp89e, Curo86, Dick86a, Dimm86, Ede189a, Fink88, Fink89, Fink90, Info88c, Jend87, Mart86, Mout88b, Ora889a, Ora889b, Ora889c Ora889d, Ora889e, Ora889f, Ora889g, OraC___, OraC84a, OraC84b, OraC84b, OraC84c, OraC84d, OraC84e, OraC84f, OraC85, OraC86, OraC87, OraC88, OraC89, Rose87a, Rose87b, Roti90, Sach87, Shaw88, Taur88a, Wong88, Zuss88].

é um SGBD baseado no modelo relacional, desenvolvido pela Oracle Corporation, representado e comercializado no Brasil na área de microcomputadores pela Compucenter Informática Ltda. e na área de mini e mainframes pela Oracle do Brasil.

a. Estrutura física e organização do espaço de armazenamento

Um banco de dados ORACLE consiste de um arquivo de banco de dados composto de partições e arquivos. Partições são unidades lógicas e arquivos são unidades físicas. Cada partição corresponde a no mínimo um arquivo físico. O número máximo é limitado pelo sistema operacional.

A unidade básica de armazenamento de dados é a tabela. Sua criação implica na reserva automática de espaços de armazenamento (segmentos) para dados e índices. Segmentos são liberados e neutralizados após a deleção da tabela.

Os dados e o dicionário de dados são armazenados em tabelas de duas dimensões. O acesso aos dados é feito especificando-se o nome das linhas; ou colunas que funcionam como índices. Para representar corretamente os dados em tabelas e facilitar a recuperação futura dos mesmos, devem ser definidas relações otimizadas com chave primária e, caso haja necessidade, chaves estrangeiras. O endereçamento para as tabelas é direto e as chaves originam índices de acesso.

O número máximo de colunas por tabela é limitado a 254, e não há limite para o número de tabelas e linhas por tabela. Campos são armazenados suprimindo os nulos (brancos e zeros não significativos) para reduzir o espaço de armazenamento.

A transferência de dados entre a memória principal e o disco é feita em blocos de registros. Visões são tabelas virtuais utilizadas para representar o mesmo dado de forma diferente para usuários diferentes. Estas tabelas não contêm dados, mas ponteiros para os dados reais.

Os dados são acessados/armazenados por blocos e não por registros, fazendo uso das técnicas de concentração e indexação. Assim, reduz-se o número de E/S.

A concentração é controlada pela especificação das colunas cujos dados são frequentemente compartilhados entre tabelas (chave de "cluster"). É gerado, automaticamente, um índice sobre as colunas que compõem a chave de "cluster". Resulta, com isso, consegue-se reduzir o espaço de armazenamento e a tempo de busca, pois dados comuns estão armazenados uma única vez e dados relacionados compartilham espaço em disco adjacentes.

Não deve-se usar a concentração se as tabelas são voláteis. Nestes casos, o SGBD dispõe de uma técnica de indexação própria baseada no conceito de árvore B+ balanceada.

Usuários podem criar índices e alocar dados de múltiplas tabelas na mesma página de base de dados, para melhorar o desempenho das consultas e otimizar o acesso aos dados em disco. índices são armazenados usando árvore B+ balanceada. Não há limite para o número de índices por tabela, porém, não devem ser criados muitos índices para evitar aumentar a improdutividade causada por atualizações na tabela.

Solicitações SQL são analisadas pelo otimizador de acessos que através de algoritmos adaptativos seleciona o "melhor" caminho de acesso. A partir da inicialização do sistema, os algoritmos básicos são continuamente adaptados, de modo a atender os comandos no menor espaço de tempo.

O controle da utilização do espaço em disco é feito pelo CCF ("Create Contiguous File"), permitindo ao usuário administrar à criação de arquivos contíguos. Os registros da log a a recuperação dinâmica são armazenados de forma comprimida. Os espaços liberados são reutilizados.

b. Tipos de dados

Os tipos de dados suportados pelo ORACLE são:

i. Char

Cadeia de caracteres de comprimento variável limitada a 240 caracteres. Cada caracter ocupa 1(um) byte.

ii.Number

Valores numéricos com/sem sinal com precisão de até 38 dígitos, armazenados com comprimento variável em até 19 bytes. Cada byte contém 2 dígitos.

iii.Long

Cadeia de caracteres de comprimento variável limitada a 65.536 caracteres, não permitindo indexação. é usada para caracterizar colunas e cada tabela pode ter apenas uma coluna do tipo "long"

iv.Raw e Long Raw

Similares aos tipos "char" e "long", respectivamente. Porém, ignora-se o significado dos bytes. Usa-se para representar dados binários e cadeias de bytes. (Ex: sequências de caracteres gráficos).

v. Date

Datas no formato especificado pelo usuário, ocupando 7 bytes de espaço de armazenamento.

c. Dicionária de dados

Definições de dados de aplicações são armazenadas e documentadas no dicionário de dados (CASE*DICTIONARY) para garantir integridade e consistência aos dados gerados em todas as fases do desenvolvimento da aplicação. A estrutura do banco de dados, também, é definida no dicionário de dados para permitir que todas as aplicações usem as mesmas

definições. Modificações são realizadas em um único local e propagadas para todas as aplicações que as utilizam.

O dicionário de dados é criado e mantido por ferramentas de CASE ("Computer Assisted Systems Engineering") que permitem acessos ao banco de dados e fornecem uma metodologia estruturada para projetar e implementar aplicações. Estas ferramentas são: o CASE*DICTIONARY, descrito anteriormente, o CASE*METHOD para definir informações a serem coletadas, e o CASE*DESIGNER com interface gráfica para o dicionário de dados permitindo criar e manipular diagramas de CASE.

d. Linguagem de definição e manipulação de dados

Uma variação da linguagem relacional SQL sintaxe IBM (SQL*PLUS), garanta a recuperação, manipulação, definição e integridade dos dados. A estrutura das tabelas e as visões lógicas, associadas a cada tabela, são definidas e manipuladas usando a linguagem SQL.

Pode-se usar o editor de textos do ORACLE ou outro editor para criar programas. Palavras reservadas não podem ser usadas como nomes de tabelas, campos ou variáveis. Não há posição fixa dos comandos na linha e no número de comandos por linha, podendo ser delimitados por vários brancos.

Telas de auxílio ajudam o usuário na sintaxe e uso dos comandos, na codificação de programas e depuração de erros.

Pode-se usar extensões do SQL para realizar consultas "ad hoc", conversão de tipos de dados, execução de consultas

aninhadas, controle de acesso e formatação de relatórios para análise de dados. Estas extensões encontram-se incorporadas a linguagem SQL*PLUS.

A entrada de dados e comandos pode ser feita em massa via leitura de arquivos externos ou interativamente por menus via tela, que deve conter os campos do arquivo a serem atualizados. Cada arquivo pode estar associado à várias telas e vice-versa.

Telas de entrada de dados podem possuir regras de validação associadas aos campos para validar dados fornecidos pelo usuário quanto a autorização de acesso ao campo, conteúdo, formato e comprimento, e gatilhos ("triggers") para serem executados quando ocorrerem determinados eventos a nível de campo, arquivo e tela. Gatilhos podem conter comandos SQL e procedimentos de validação a serem executados durante a entrada de dados, para validar entradas e controlar o fluxo de execução de uma aplicação. Regras e gatilhos são armazenados no dicionário de dados e associados aos campos no arquivo, não precisando ser especificados a cada tela.

A linguagem SQL*PLUS dispõe, também, de comandos para a emissão de relatórios. A especificação de caracteres de formatação, adapta o relatório ao formato desejado.

e. Interface com usuário

Uma interface interativa por menus permite construir e usar a base de dados via aplicativos não procedurais baseados em telas de entrada de dados, sem precisar codificar uma

aplicação, isto é, sem programação. Dentre outras funções, os menus pré-definidos possuem opções para construir telas, consultas, relatórios e gráficos, selecionar e atualizar dados e tabelas, gerenciar bancos de dados, migrar dados entre produtos e máquinas diferentes, gerar interfaces por aplicação e estabelecer níveis de segurança. As instruções selecionadas são combinadas com informações do dicionário de dados e geram os objetos de uma aplicação.

Interfaces permitem acessar o banco de dados através de outros produtos. A interface ORACLE 123 permite ao usuário acessar/fornecer dados do ORACLE diretamente do Lotus 123.

Há outras interfaces para linguagens C, Cobol, Fortran, PL/1, Pascal e Ada, e uma interface gráfica para a geração de gráficos sobre os dados do banco de dados.

f. Integração com outros SGBD's

Aplicações desenvolvidas em SQL/DS e DB2 [IBM_85, IBM_87, Kahn84, Mudi88, Rag189] e em outros SGBD's que utilizam a linguagem SQL no padrão IBM, rodam sem alterações no ORACLE em qualquer ambiente e vice-versa.

Dados armazenados em outros SGBD's podem ser convertidos em formato ORACLE via SQL*LOADER. Assim, permite-se migrar dados entre bancos de dados diferentes, ler dados de arquivos externos ao sistema, carregar tabelas a partir de dados previamente armazenados e usar hardwares diferentes para um único banco de dados lógico. Porém, a carga de tabelas requer que estas existam no momento da carga.

g. Qualidade da documentação

A documentação disponível é muito vasta. O suporte ao SGBD é dado por representantes nacionais com alto conhecimento do produto. Porém, como os demais produtos estrangeiros, possui manuais, mensagens, comandos e telas em inglês, dificultando a compreensão de usuários leigos neste idioma.

Para usar o SGBD de forma correta e otimizada, aproveitando ao máximo seus recursos, deve-se investir em treinamento. O ORACLE é complexo, e com algum grau de dificuldade no uso das ferramentas. Precisa-se conhecer as potencialidades e deficiências do produto para melhor usá-lo.

h. Segurança e proteção dos dados

A monitoração do uso dos bancos de dados e a deteção de tentativas de violação das normas de segurança podem ser obtidas pela ferramenta SAF ("Security Audit Facilit"). É possível estabelecer níveis de segurança para a aplicação e realizar auditorias no ORACLE, reconstruindo os caminhos de utilização do sistema de forma a detectar tentativas de uso indevido e analisar padrões de utilização.

Usuários podem, comentar, acessar dados e realizar operações as quais estejam autorizadas para ele. Logo, são definidas senhas e níveis de acesso aos dados, que ficam armazenados no dicionário de dados. Podem ser estabelecidas auditorias a nível de arquivo, dicionário de dados e sessão.

i. Deteção de Falhas e reconstrução do banco de dados

O mecanismo recuperador de dados JI ("Journal Image"), em falhas de hardware e sistema, anula todas as transações incompletas e garante a consistência do banco de dados.

Transações com atualização ativam o JI que copia uma imagem dos dados em arquivo auxiliar BI ("Before Image File"), único por sistema. Atualizações são feitas no BI e tornam-se permanentes no banco de dados após as transações estarem completas e consistentes. Falhas de transação ocorridas durante a execução da mesma, não danificam o banco de dados, que conserva a versão que iniciou a transação.

Adicionalmente, todas as transações de atualização do banco de dados executadas com sucesso são armazenadas em arquivo auxiliar AI ("After Image File"). Caso ocorram falhas de meio, deve-se reexecutá-las sobre cópia do banco de dados previamente gerada, recriando o estado corrente dos dados. Para tanto, é necessário que se gere, periodicamente, cópia dos dados e recupere-a quando necessário. O utilitário EXPORT gera cópia em arquivo no sistema operacional e o IMPORT recupera e recarrega o arquivo no banco de dados.

j. Concorrência

Em ambientes multiusuário, pode-se compartilhar programas e dados frequentemente usados, reduzindo as operações de E/S realizadas pelo SGBD e o espaço em disco necessário para

armazenar as informações, apesar de aumentar o número de usuários concorrentes na instalação. Mecanismos de bloqueio garantem integridade aos dados e controle de concorrência. Durante atualizações em um arquivo, este permanece protegido e inacessável por outro usuário. Não é permitido que dois usuários atualizem o mesmo dado ao mesmo tempo.

Há vários tipos de bloqueio. São eles:

i. Modo exclusivo

Bloqueia os dados para atualizações por outros usuários durante atualizações nos mesmos e impede o estabelecimento de outros bloqueios.

ii.Modo compartilhado

Bloqueia os dados para atualizações por outros usuários durante leituras.

iii.Modo compartilhado atualizado

Reserva o direito de atualizações futuras em linhas de tabelas. Durante a atualização, as linhas ficam bloqueadas para outras atualizações, entretanto durante a reserva, estas podem ser consultadas e bloqueadas para atualizações por outros usuários.

Outros mecanismos bloqueiam o dicionário de dados a nível de operação, definição e tabelas. Os bloqueios permanecem até que as atualizações estejam permanentes no banco de dados, sejam abortadas pelo usuário ou ocorram falhas.

Caso ocorra "deadlock", aborta-se a transação que atualizou menos o banco de dados, isto é, a mais incompleta.

k, Ambiente Operacional

Em ambientes monousuário que usem microcomputadores do tipo PC, o SGBD necessita na sua versão 2.0, de uma configuração mínima de hardware de 640 Kb de memória RAM e 5 Mb em disco rígido. As versões posteriores necessitam de 640 Kb de memória RAM, 1 Mb de memória extendida, 8Mb em disco rígido, microcomputador do tipo PC-AT e sistema operacional MS-DOS versão 3.1 ou posterior.

A instalação do SGBD envolve a especificação de parâmetros, que determinam as características do ambiente. Nesta tese, sua instalação utilizou os valores de parâmetros padrão, de forma a testar seu desempenho num ambiente "default".

1. Portabilidade e flexibilidade

O ORACLE e suas aplicações rodam em ambientes mainframes IBM, superminis UNIX e microcomputadores PC's. Pode-se migrar aplicações entre máquinas. Suas características são padrões aos sistemas operacionais VM/CMS, MVS, UNIX, XENIX, VMS e MS/DOS. O acesso à máquinas diferentes com sistemas operacionais diferentes é feito pela SQL*CONNECT.

O ORACLE é um SGBD distribuído que permite aos usuários acessar dados armazenados numa rede de computadores

diferentes através do SQL*NET. Os dados são tratados como se estivessem armazenados num mesmo computador.

Para manter a portabilidade e flexibilidade do ORACLE, há correspondência entre os tipos de dados ORACLE e aqueles dos SGBD's IBM. Na conversão de aplicações, tipos IBM são convertidos, automaticamente, para seu tipo equivalente.

O SGBD permite conectar diversos hardwares ao mesmo tempo. Dados do banco de dados localizado no mainframe podem ser acessados por PC's e os resultados de atualizações nestes dados devolvidos ao mainframe. Organizações podem integrar dados gerenciados por SGBD's diferentes através de uma rede de informações unificadas [OraC88].

m. Gerador de Aplicações

Aplicações podem ser desenvolvidas por usuários finais via Gerador de Aplicações (EASY*SQL), com esforço e custo de desenvolvimento reduzidos. Pode-se fazer consultas, gerar relatórios e gráficos, criar e atualizar dados, tabelas e visões, gerenciar bases de dados e transferir dados entre produtos e equipamentos, selecionando opções em menus.

Pode-se construir aplicações via prototipação. É gerada uma versão preliminar do sistema que vai sendo refinada de forma a incorporar novas características e necessidades, e a eliminar erros. A versão final da aplicação é uma próxima da realidade do usuário e pode ser implantada de forma imediata. A geração das versões é feita pelo EASY*SQL.

O Gerador de Telas (SQL*FORMS) permite a criação de telas otimizadas para operar com o banco de dados. Os campos são dispostos de modo a facilitar a visualização dos mesmos. Inserções, deleções, atualizações e consultas a registros podem ser feitas em telas formatadas que dispensam o uso da linguagem SQL, e não impõem limite para o número de páginas e arquivos que as compõem.

O Gerador de Relatórios (SQL*REPORT) permite a criação de relatórios automaticamente, e o controle da formatação. Compõe-se de dois módulos: RPG ("Report Generator") e RPF ("Report Formatter"). O SQL*REPORT é não procedural e dirigido por menus, podendo ser usado, também, como processador de textos e para emissão de mala direta.

O Gerador de Gráficos (SQL*GRAPH) converte resultados de consultas no gráfico selecionado pelo usuário. Estão disponíveis vários formatos de gráficos.

61 Gerador de Meriui (SQL*MENUS) peimite ao usuário gerar mentis disiâmicos para aplicaçães. Estes são conti-olados p(ri-2abelas DRACLE, Deste modo, pode-se criar e mantei sisti maf; comp1etos, alterarido-a ; apesias ar; esi2i-adas em uma tabela.

n. Arquitetura

Sua arquitetura aberta (SQL*STAR) une bancos de dados distribuídos em um banco de dados lógico, independente destes estarem localizados em computadores diferentes. Assim, combinando-se o ORACLE e alguns utilitários, pode-se integrar computadores e sistemas operacionais diferentes,

além de SGBD's diferentes que utilizem o padrão relacional. Usuários podem acessar dados de qualquer computador, mesmo sem conhecer a localização dos dados.

A arquitetura é composta de três módulos principais: uma máquina de gerência de dados relacional (ORACLE KERNEL) e um conjunto de ferramentas (CCF, EASY*SQL, TPSS, SAF, PRO*C PRO*COBOL, SQL*CALC, SQL*DBA, SQL*FORM, SQL*GRAPH, SQL*NET, SQL*LOADER, SQL*MENUS, SQL*PLUS, SQL*QMX e SQL*REPORT).

o. Utilitários

O usuário final pode interagir com o banco de dados, tendo acesso as informações através de ferramentas integradas de suporte (gráficos, planilhas e formulários). Pode-se consultar a base de dados por menus similares aos usados pelo QMF do DB2/IBM, combinando-se os recursos dinâmicos de consulta com um gerador de relatórios (SQL*QMX).

O desempenho procedural da SQL*PLUS pode ser otimizado usando-se o TPSS ("Transaction Processing SubSystem"). Este permite o processamento de transações on-line (OLTP), tolerância a falhas e atualizações em alta velocidade.

Programas e subrotinas em C, Cobol, Fortran, PL/1, Pascal e Ada podem ser integrados a aplicações ORACLE usando pré-compiladores (PRO*C, PRO*COBOL, PRO*FORTRAN, PRO*PL/1, PRO*PASCAL e PRO*ADA, respectivamente). Usuários podem, também, desenvolver aplicações nestas linguagens e acessar os dados do ORACLE via comandos destes pré-compiladores.

O SQL*DBA é uma ferramenta interativa de auxílio ao administrador de banco de dados. A monitoração do uso do SGBD e do desempenho de transações, e a geração de estatísticas de desempenho do ORACLE é realizada pelo módulo ODS ("Oracle Display System Utility"). A ativação do processo de recuperação do banco de dados é feita pelo AIJ ("After Image Journalling Utility"). Finalmente, a ativação /desativação do ORACLE é tarefa do IOR ("Init Oracle").

O SQL*CALC é uma planilha eletrônica integrada ao ORACLE que permite transformar planilhas em bancos de dados e aplicar comandos SQL ao conteúdo das planilhas. Através da interface "ORACLE 123" é possível carregar planilhas desenvolvidas em Lotus para o SQL*CALC. Entretanto, este utilitário não possui comandos gráficos.

O SQL*NET transforma os bancos de dados da instalação em distribuídos, gerenciando o ambiente como um todo. Porém, deve ser usado em ambientes multiusuário DECNET.

Finalmente, existe um sistema de correio eletrônico e ferramentas de automação de escritório.

A.5 PARADOX

A seguir, será apresentada uma descrição do PARADOX. Informações podem ser obtidas em [Ansa87a, Ansa87b, Ansa87c, Ansa88a, Ansa88b, Ansa88c, Ansa88d, Ansa88e, Ansa89, Bar86, Bor189, Bot290, Coi88, Crab87, Curo86, Demo86, Derf88, Dick86a, Ede189a, Fast89, Gold90, Info88d,

Inte___, Inte88, Inte89b, John90, Jr.F86, Mias88, Mias89, Petr87, Robi89, Seym88c, Shaw88, Shaw89, Soua88, Ursc88].

é um SGBD relacional desenvolvido pela Borland International e, representado e comercializado no Brasil pela Intercorp do Brasil Ltda.

a. Estrutura física e organização do espaço de armazenamento

Os dados, resultados de busca a formatos de relatório são armazenados em tabelas de duas dimensões organizadas em linhas (registros) e colunas (campos). A estrutura das tabelas fica armazenada numa tabela especial. Porém, como há tabelas temporárias e permanentes, é necessário tornar as informações das permanentes disponíveis às temporárias.

Cada tabela pode possuir objetos associados como índices primário e secundários, telas, relatórios, e regras de validação. A tabela e seus objetos constituem uma família, sendo identificados por nomes iguais e extensões diferentes. Cada objeto é armazenado em um arquivo (tabela) em disco, embora não seja tratado como tal pelo PARADOX, por possuir características especiais.

Pode-se manipular várias tabelas ao mesmo tempo. Alterações na estrutura de uma geram, automaticamente, alterações em todas as tabelas, formulários, relatórios, programas e aplicações que a usam, e uma tabela temporária com os dados originais. Para não perder os dados alterados e, permitir auditorias futuras nas aplicações e restaurações na base de

dados, pode-se tornar as temporárias em permanentes, para impedir que sejam deletadas ao fim da sessão ou após a criação de outra com mesmo nome.

O número de tabelas, em uso concorrentemente, é limitado a capacidade do hardware da instalação. Cada tabela pode ter até 2 bilhões de registros com até 255 campos (4.000 bytes) por registro. Cada campo fica limitado a 255 caracteres.

As operações relacionais sobre tabelas ignoram o nome dos campos, considerando apenas o domínio ao qual pertencem e a ordenação destes na tabela. Assim, tabelas que possuem os mesmos campos ordenados diferentemente, podem ser comparadas erradamente e gerar como resultado tabelas incorretas. Campos com mesmo nome e localizados em posições diferentes nas duas tabelas não são comparados.

Deleções em registros não liberam, automaticamente, o espaço em disco. Cabe ao administrador do banco de dados reorganizar o espaço, para liberar áreas não mais ocupadas.

Visões são tabelas virtuais que especificam campos a serem usados por determinadas operações ou usuários, não contendo dados, apenas ponteiros para os dados reais. Usuários podem criar visões grupando campos de várias tabelas, de forma a simular a operação de junção da álgebra relacional.

Macros podem ser criadas a partir do armazenamento de comandos, execuções, ou isso interativo. Historicamente, guarda-se ações realizadas durante uma sessão num arquivo. Assim, consegue-se automatizar ações. Porém, este tipo de arquivo não pode conter alguns comandos de

programação, não devendo estes serem executados durante o intervalo da armazenamento histórico na sessão.

O método de acesso/armazenamento usado é a identificação. São definidos índices primários e secundários para acessar e identificar os dados (uma chave primária e várias chaves secundárias por tabela). Os índices são usados e mantidos atualizados automaticamente, e podem acelerar consultas. As tabelas são mantidas ordenadas pela chave primária.

Pode-se otimizar consultas frequentes, criando um índice secundário para cada campo não chave na consulta associado a um critério de seleção. Os índices são atualizados, automaticamente, e aceleram o acesso aos dados das tabelas. Porém, não deve-se criar vários índices pois ocupam espaço em disco e gastam tempo para serem atualizados.

Um gerenciador de memória virtual VMM ("Virtual Memory Management") gerencia, automaticamente, o espaço em disco e a memória disponível para otimizar o desempenho. De forma a acelerar o processamento de operações e reduzira frequência de acessos a disco, é permitido o uso de dispositivos de memória expandida. Estes aumentam a quantidade de memória principal disponível, permitindo manter mais informações na memória. O objetivo é tentar realizar todas as tarefas na memória principal com um mínimo de acessos a disco.

A disponibilidade de memória expandida permite criar uma área de armazenamento temporária e um cache de disco. A área armazena arquivos de sistema e procedimentos PAL mais recentemente usados. O cache guarda informações mais recentemente acessadas do disco ou nele gravadas,

espacialmente dados armazenados em tabelas. A alocação da memória é feita, automaticamente, entre o VMM, a área de armazenamento temporária e o cache de disco.

Porém, quando a quantidade de memória RAM disponível não é suficiente para manter um arquivo na memória, um mecanismo de memória virtual desvia dados para arquivos temporários em disco. Assim, pode-se trabalhar com arquivos grandes, embora gastem, temporariamente, mais espaço em disco. Este espaço em disco deve ser três vezes o tamanho do arquivo.

A linguagem conversacional QBE usa a técnica de otimização de consultas por heurística, da área de inteligência artificial, para otimizar acessos aos dados. O objetivo é encontrar o meio mais rápido de acessar dados, estabelendo-se índices e relacionamentos entre eles. Assim, é analisado tamanho das tabelas, número de campos chave, ordem das operações na consulta e tipo de consulta. A ligação entre arquivos é feita por campos comuns, não sendo necessário que estes campos sejam chave ou tenham o mesmo nome.

b. Tipos de dados

Os tipos de dados suportados pelo PARADOX são:

i. A (Caracter)

Cadeia de caracteres alfanuméricos com comprimento máximo de 255 caracteres.

ii.S (Numérico inteiro)

Valores inteiros com/sem sinal entre -32.767 e 32.767. Não podem ser utilizados em opções de entrada de dados e formatação de apresentação.

iii.N (Numérico decimal)

Valores numéricos com/sem sinal com precisão de até 15 dígitos significativos.

iv.$(Monetária)

Valores numéricos monetários com características iguais ao tipo N, e formato de apresentação diferente (arredondamento para duas casas decimais, separadores de números inteiros e parênteses para os negativos).

v. D(Data)

Datas no formato especificado pelo usuário, entre 1 de janeiro de 100 e 31 de dezembro de 9999.

c. Dicionário de dados

Armazena informações sobre índices secundários, relatórios e telas, e consistências de dados associadas à cada tabela. Atualizações nas tabelas e dados são, automaticamente, incorporadas no dicionário e nos objetos que as usam.

d. Linguagem de definição e manipulação de dados

Uma linguagem de programação, denominada PAL ("Paradox Application Language"), permite desenvolver aplicações em

ambientes mono e multiusuário. Estão disponíveis comandos para controlar a impressão, construir telas de entrada de dados e relatórios de saída, acessar e atualizar o banco de dados, e desenvolver consultas aninhadas, não sendo preciso desmembrá-las em vários comandos. Porém, alguns comandos disponíveis no modo interativo, não o estão na PAL, limitando o escopo de programas que se pode desenvolver.

Suas palavras reservadas não podem ser usadas como nomes de tabelas, campos ou variáveis. Não há posição fixa dos comandos na linha e no número de comandos por linha, sendo possível delimitá-los por vários brancos. Pode-se usar o editor de textos PSE ("Paradox Script Editor") ou qualquer outro editor o qual o usuário esteja familiarizado.

Telas de auxílio ajudam o usuário na sintaxe e uso dos comandos, na codificação de programas e depuração de erros, e possuem um índice de auxílio por tópico.

A entrada de dados e comandos pode ser feita em massa via leitura de arquivos externos ou via tela que deve conter os campos do arquivo a serem atualizados. Cada arquivo pode estar associado à várias telas e uma tela pode realizar a entrada de dados em vários arquivos, simultaneamente.

Uma tela de entrada de dados padrão é criada para todo arquivo gerado pelo SGBD, automaticamente. Pode-se alterá-la e definir 14 outras ligadas ao arquivo. Estas suportam a definição de regras de validação associadas aos campos para a conferência dos valores fornecidos pelo usuário quanto a formato, comprimento e conteúdo. Pode-se, também, criar

ligações entre tabelas para garantir integridade referencial aos dados durante operações no banco de dados.

As regras ficam associadas aos campos no arquivo físico e armazenadas do dicionário de dados. Logo, não precisam ser especificadas a cada tela e, entradas de dados realizadas nos arquivos são criticadas, automaticamente. As regras permitem realizar consultas à várias tabelas para validar os dados, porém, não pode-se incluir comandos PAL na sua especificação. Procedimentos de validação adicionais devem ser embutidos nos programas que ativam as telas, para validar os dados antes da atualização do arquivo.

Da mesma forma que na entrada de dados, cada arquivo criado gera, automaticamente, um relatório e um formulário padrão de saída de dados. O relatório grupa vários registros de uma tabela enquanto o formulário trabalha com um registro de cada vez. Ambos podem ser alterados pelo usuário, interativamente, que pode, ainda, definir 14 outros relatórios e 14 outros formulários no formato especificado. A especificação de caracteres de formatação pelo usuário adapta o relatório ao formato desejado e permite a emissão de etiquetas, cartas e formulários padrões.

A linguagem suporta, também, a chamada de rotinas em C e Pascal, e procedimentos e macros desenvolvidos em PAL.

Pode-se acessar os dados do banco de dados utilizando-se a linguagem conversacional QBE ("Query-By-Example"), que permite consultar, definir e atualizar informações em uma ou mais tabelas sem conhecimento da sintaxe dos comandos. A linguagem visual e intuitiva dispensa programação e permite

buscas a valores similares aos especificados na consulta, de forma a detectar erros de datilografia e informações imprecisas fornecidas pelo usuário.

e. Interface com usuário

Uma interface interativa por menus permite navegar entre telas para a seleção de comandos. O usuário não precisa conhecer a sintaxe dos comandos para selecionar uma função.

As telas estão divididas em 3 áreas. A primeira é a área de menu, que mostra opções de comando e mensagens de situação com informações sobre operação e objeto corrente. A segunda é a área de trabalho, onde estão as tabelas e formulários de trabalho. A terceira é a área de mensagens onde são apresentados avisos de erros e advertências ao usuário.

Interfaces permitem acessar/fornecer dados de/para arquivos tipo WKS (Lotus 123), DBF (dBase II e III), DIF (VisiCalc), PFS (IBM Professional Files), SYL (Multiplan) e ASCII.

Há outras interfaces (API - "Application Programming Interfaces") para linguagens C e Pascal, e para a geração de gráficos sobre os dados do banco de dados.

f. Integração com outros SGBD's

A importação/exportação de dados entre bancos de dados é feita no formato ASCII, WKS, DBF, DIF e PFS. O usuário não precisa conhecer a estrutura do arquivo e pode alterá-lo

antes do término da operação. Entretanto, não há integração entre o PARADOX e outros SGBD's.

g. Qualidade da documentação

A documentação disponível é suficiente. O suporte ao SGBD é dado por representantes nacionais. Possui alguns manuais, mensagens, editados e telas em inglês, dificultando a compreensão de usuários leigos neste idioma. Contudo, a interface interativa por menus principal (associada a linguagem PAL) possui comandos, manuais, mensagens e telas em português, facilitando acesso ao SGBD.

, Segurança e proteção dos dados

A proteção aos dados é feita pela definição de múltiplas senhas que os protegem a nível de usuário, objeto, campo, tabela e família. Apenas usuários autorizados podem acessá-los. Usa-se a criptografia para proteger dados específicos.

Pode-se definir níveis de segurança pelo PPG ("Paradox Protection Generator") ou por comandos da linguagem PAL. São gerados níveis de acesso para os objetos PARADOX.

i. Deteção de Falhas e reconstrução do banco de dados

Não há um mecanismo restaurador automático para restaurar totalmente os dados em caso de falhas de meio. Devem ser feitas cópias de segurança antes de iniciar cada sessão, de

forma a garantir o retorno do banco de dados a um estado consistente após a ocorrência de falhas.

Porém, o "TUTILITY" permite restaurar tabelas danificadas por falhas de hardware e quedas de energia. Uma interface por menus orienta na escolha dos comandos. A tabela é analisada para detectar falhas e, então, recuperada. Em alguns casos, não é possível a recuperação total da tabela, sendo necessário usar a cópia de segurança.

Transações de atualização tornam-se permanente no banco de dados quando estiverem completas. O VMM atualiza e controla informações na memória e em disco, mantendo as alterações na memória até o término da transação. Falhas de transação ocorridas na execução da mesma, não danificam o banco de dados, que conserva a versão que iniciou a transação.

A PAL possui i depurador que é ativado, automaticamente, quarido ocorrem falhas na execução de aplicações.

j. Concorrência

Em ambientes multiusuário, pode-se compartilhar programas e telas e, atualizar e consultar dados e gerar relatórios de tabelas comuns. Um mecanismo de bloqueio garante compartilhamento aos dados e controle de concorrência, sem perda de integridade e consistência. Para tanto, durante atualizações em um arquivo, este permanece protegido e inacessível por outro usuário. Não é permitido que vários usuários atualizem um arquivo ao mesmo tempo.

No momento do acesso aos objetos pelo usuário, estes são bloqueados, automaticamente. É aplicado o bloqueio menos limitante de conformidade com a operação em execução, para permitir o maior acesso possível aos objetos por todos os usuários. Outros usuários que desejarem atualizá-lo, receberão uma mensagem informando que o objeto está em uso.

Há vários tipos de bloqueio no PARADOX para complementar o automático. São eles:

i. Bloqueio de registro

Permite que vários usuários acessem o mesmo objeto ao mesmo tempo, porém, bloqueia o registro corrente em edição durante atualizações e o desbloqueia ao término destas, tornando as modificações permanentes no banco de dados. Registros bloqueados para atualizações permanecem desbloqueados para leituras.

ii.Bloqueio Explícito

Permite manter controle integral sobre o acesso de usuários aos objetos partilhados.

iii.Bloqueio Integral

Impede acessos concomitantes ao objeto bloqueado. Deste modo, garante-se o uso exclusivo de objetos.

iv.Bloqueio contra gravação

Impede atualizações na estrutura e conteúdo dos objetos bloqueados, mas permite leituras aos mesmos.

v. Bloqueio de família

Impede atualizações na estrutura e conteúdo de todos os membros da família da tabela bloqueada, mas permite leituras aos mesmos.

Usuários podem impedir o estabelecimento de bloqueios sobre objetos, através de mecanismos anti-bloqueio. São eles:

i. **Anti-bloqueio contra gravação**

Impede o estabelecimento de bloqueios contra-gravação e integral no objeto. A qualquer momento pode-se realizar operações de atualização apenas com bloqueio automático exceto se forem estabelecidos outros bloqueios.

ii. **Anti-bloqueio integral**

Impede o estabelecimento de bloqueio integral ao objeto e fornece o nível máximo de uso concomitante permitido, a menos que sejam estabelecidos outros bloqueios.

Para manter as telas atualizadas com a última versão dos dados, há procedimentos para atualizar, automaticamente e instantaneamente, as informações em todas ("Auto-refresh"). Após a edição, telas ativas na sessão podem ser atualizadas em tempo real, em intervalos fixos ou mediante solicitação, dependendo da opção de renovação selecionada pelo usuário.

Assim, um usuário pode gerar gráficos ao mesmo tempo em que os dados estão sendo fornecidas por outro usuário, pois o PARADOX atualiza os dados em tempo real, modificando os gráficos na medida em que o usuário entra com os dados.

k. **Ambiente Operacional**

Em ambientes monousuário que usem microcomputadores do tipo PC-XT ou PC-AT, o SGBD necessita de uma configuração mínima de hardware de 512 Kb de memória RAM, 2Mb em disco rígido para o SGBD (o uso do gerador de aplicações PPP requer 2 Mb em disco rígido adicionais), e sistema operacional MS/DOS versão 2.0 ou posterior. Porém, em ambientes multiusuário, é necessário 640 Kb de memória RAM e sistema operacional MS/DOS versão 3.1 ou posterior.

1. Portabilidade e flexibilidade

O PARADOX e suas aplicações rodam em ambientes mono e multiusuário, incluindo mainframes IBM e microcomputadores PC's. Suas características são padrões a diversos sistemas operacionais como MS/DOS, VAX/VMS, IBM/MVS, IBM/VSE, OS/2, IBM/VM e IBM/386, podendo migrar aplicações entre máquinas.

m. Gerador de Aplicações

O Gerador de Aplicações PPP ("Paradox Personal Programmer") com sua interface interativa por menus, permite desenvolver a alterar aplicações orientadas a menus, sem programação. A definição dos objetos, menus e ações a estes associadas, é feita mediante seleção de comandos. Os recursos do PPP permitem definir a estrutura do banco de dados, relatórios, telas interativas de E/S, gráficos, tabelas, restrições de integridade, visões, e proteção aos dados, consultar a atualizar informações, e chamar macros previamente criadas.

Entretanto, a geração das telas, relatórios e gráficos é feita por geradores próprios associados ao PPP.

As telas do PPP são divididas em quatro setores. O primeiro contém o menu com comandos disponíveis. O segundo mostra operações realizadas e comandos a serem executados em sequência. O terceiro é um setor de auxílio que informa os diversos passos necessários a criação e alteração de uma aplicação. Finalmente, o quarto é uma janela de mensagens.

Após a seleção das tarefas a serem executadas, o PPP gera o código da aplicação em PAL e os objetos especificados. Uma documentação descritiva dos arquivos utilizados, operações associadas e árvore de comandos é produzida. Rotinas de erro padrões são, automaticamente, criadas para enviar mensagens ao usuário sempre que anomalias acontecerem. A execução das aplicações desenvolvidas com o PPP pode ser feita pelo "Paradox RunTime".

Em seguida, o usuário pode alterar o código gerado. Porém, alterações realizadas diretamente no código, serão desativadas em caso de uso posterior do PPP para modificar a aplicação, pois o PPP age sobre a última versão por ele gerada. Logo, alterações em aplicações geradas pelo PPP devem ser feitas utilizando o PPP, para evitar perdas de informações e inconsistências entre versões.

A construção de aplicações pode ser feita por prototipação. Usa-se o PPP para gerar versões sucessivas da aplicação que vão sendo refinadas e modificadas, eliminando erros e incorporando novas características. Usuários obtêm

resultados práticos mais rapidamente e podem detectar falhas e necessidades nas fases iniciais da implementação.

Um Gerador de Telas permite desenvolver telas otimizadas, automaticamente. Este está associado ao PPP, possui uma interface interativa por menus que dispensa a programação e dispõe dos mesmos recursos das linguagens PAL e QBE.

Um Gerador de Relatórios permite a criação de relatórios automaticamente, e o controle da formatação. São guardados ponteiros para os objetos integrantes do relatório, para atualizar sua especificação, automaticamente, após mudanças nos objetos. Este está associado ao PPP, possui uma interface interativa por menus que dispensa a programação e dispõe dos mesmos recursos das linguagens PAL e QBE.

O Gerador de Gráficos ("QUATTRO") associado ao PPP, permite criar vários tipos de gráficos a partir das informações do banco de dados. Assim, dados normalizados são convertidos para um formato compatível com o utilitário via "CROSSTAB". A interface interatira por menus permite ao usuário criar, alterar, e exportar gráficos sem programação.

Aplicações desenvolvidas pelo PPP sofrem restrições que se podc superiai pela ci ctrução dE pr.ocedimentoc em PAL, isicoi-porai;ão destes ao PPP E a4ivação doc; mesmos pelos menus. Em opaiações multiusuário, não ce pode bloquear/ desbloquear tabelas e registros explicitamente, e realizar mapeamentos de relacionamesitoc um-para-muitoc;. Cada aplicação fica limitada a utilizar até 15 tabelai; e, possuir a4é 15 seleções por meniu e no máximo 15 níveis de menu. A con84r.ui;ão de apiciai;ões obriga a existëmcia de uni

menu par tela da aplicação e segue a padrão de telas da interface do PARADOX com subdivisões em setores.

Não se pode gerar aplicações pelo PPP sem o uso de menus. Logo, torna-se inviável sua aplicabilidade a esta tese que fará a entrada de dados e comandos das aplicações em massa via leitura de arquivos externos sem o uso de telas, para eliminar a improdutividade gasta com a seleção de opções em menus e, digitação e fornecimento de informações via tela.

n. Arquitetura

Não foi possível obter informações sobre a arquitetura do SGBD. A documantação disponível não relata ditalhes de implementação e da arquitetura utilizada pelo PARADOX,

o. Utilitários

O "Paradox RunTime" armazena aplicações desenvolvidas no PARADOX para executá-las em ambientes que não o utilizam.

Aplicações desenvolvidas em PAL sob uma determinada versão do PARADOX podem ser traduzidas para outras versões através do "PAL TRANSLATOR" [Inte88], independente do idioma usado pela linguagem (exemplo: inglês --> francês). A tradução é feita pela substituição de itens de texto específicos da versão original (comandos, palavras-chave, nomes de tabelas temporárias, e cadeias de caracteres usadas pela PAL ou pelo SGBD). O texto resultante necessita ser editado quando os objetos precisam ser alterados para usar o idioma

e a sintaxe de comandos corrente, e facilitar a legibilidade e manutenibilidade das aplicações. Pode-se, também, extender o dicionário de dados padrão do SGBD, incorporando novos argumentos e palavras reservadas, reduzindo o esforço de tradução manual.

Aplicações em C podem masipular dados e tabelas, e utilizar furiçõec disponíveis no PARADOX pelo PE ("Pavadox Esigine") [Roti90]. Asciii, aplitações desarivolvidas em PAL podeti aei- ::: em: i:: c . c: s:: código . Para tanto, o P& fornets todac as funções (rotirias) necessárias para que um programa em C acesi e o banco de dadoc, e uma interface API ("Application Programmirig Isiterface") que permite a inPegração entre o C e o PARADOX. A chamada das funções é feita pelo programa C.

p. Outros (Extensão do PARADOX ao padrão SQL)

Segundo [Mias89], a linguagem SQL estará disponível na próxima versão do PARADOX (3.0). Haverá um utilitário para traduzir consultas QBE e comandos PAL em consultas SQL.

O Geradcr SQL ear2ará pveserite sia vevção 3.0 c permi2irá imp;rtai-/cxpovtar dados entre o PARADOX e um SGBD isisPalado no mainframe que usa SQL rio padrão IBM. Além dicso, disporá da 1isiguagem de cosisulta SQL (PC/SQL/LINK).

A.6 RBASE SYSTEM V

A seguir, será apresentada uma descrição do RBASE SYSTEM V. Informações podem ser obtidas em [Bott90, Cobb87, Corr88,

Curo86, Derf88, Dick86a, Micr86a, Micr86b, Micr86c, Micr86d, Micr86e, Petr87, Poor85a, Poor87, Rubk87, Seym88c, Soua88].

é um SGBD relacional desenvolvido pela Microsoft Corporation e, representado e comercializado no Brasil pela Compucenter Informática.

a. Estrutura física e organização do espaço de armazenamento

Um banco de dados RBASE SYSTEM V consiste de 3 arquivos em disco. Cada arquivo possui um número de identificação próprio e uma extensão única. O primeiro contém a definição da estrutura do banco de dados (esquema) e a localização dos dados. O segundo contém os dados propriamente ditos, enquanto o terceiro é o arquivo de índices.

Estes arquivos devem permanecer juntos em disco. Acessos ao banco de dados ativam os três arquivos. Tentativas de atualização nestes via outros produtos danificam os mesmos.

Os dados são armazenados em tabelas de duas dimensões organizadas em linhas (registros) e colunas (campos). Cada categoria de dados (entidade) origina uma tabela que contém dados desta entidade. Pode-se definir relacionamentos entre tabelas através da especificação dos campos que os compõem.

Os objetos associados com cada entidade ficam, também, armazenados em tabelas do banco de dados. São eles: telas,

relatórios, regras de validação, visões e colunas resultantes de operações sobre outras colunas.

Certas operações geram tabelas temporárias na memória. Para não perder os dados gerados e permitir auditorias futuras nas aplicações e restaurações na base de dados, pode-se tornar as temporárias em permanentes, impedindo deleções ao fim da sessão ou após a criação de outra com mesmo nome.

Cada banco de dados em ambientes monousuário pode ter até 80 tabelas diferentes e 800 atributos. A capacidade de cada registro é de 4096 caracteres. O número de registros por tabela fica limitado pela capacidade de armazenamento do disco e pelos limites impostos pelo sistema operacional.

Deleções em registros não liberam, automaticamente, o espaço em disco. Cabe ao administrador do banco de dados reorganizar o espaço, para liberar áreas não mais usadas.

Visões são tabelas virtuais que especificam campos a serem usados por determinadas operações ou usuários, não contendo dados, apenas ponteiros para os dados reais. Usuários podem criar visões grupando campos de até 5 tabelas, de forma a simular a operação de junção da álgebra relacional.

Pode-se criar arquivos EXEC pelo armazenamento de comandos executados no modo interativo. Assim, deve-se informar ao RBASE para guardar, historicamente, as operações realizadas durante a sessão em arquivo. Assim, consegue-se automatizar ações. Porém, este tipo de arquivo não pode conter alguns comandos de programação, não devendo estes serem executados durante o intervalo de armazenamento histórico na sessão.

Os métodos de acesso/armazenamento usados são a indexação por árvore B+ e a identificação por chave. Colunas marcadas como chave são indexadas e usadas para localizar registros no banco de dados, pois índices guardam a localização dos dados e, as buscas a informações são baseadas nos índices.

b. Tipos de dados

Os tipos de dados suportados pelo RBASE SYSTEM V são:

i. Text

Cadeia de caracteres alfanuméricos com comprimento padrão de 8 e comprimento máximo de 1.500 caracteres.

ii.Note

Cadeia de caracteres alfanuméricos em formato texto com comprimento máximo de 4.092 caracteres por item de dados. O "overhead" envolvido na manipulação deste tipo é de aproximadamente 4.050 caracteres.

iii.Integer

Valores inteiros com/sem sinal entre -999.999.999 e +999.999.999. Não podem ser utilizados em opções de entrada de dados e formatação de apresentação.

iv.Real

Valores numéricos com/sem sinal entre -9 X (10 ** -37) e +9 X (10 ** - 37), com precisão decimal de até 6 dígitos

e utilização da notação científica para representar números com mais de 6 dígitos.

v. Double

Valores numéricos com/sem sinal entre -1Ø X (1Ø ** -3Ø8) e +1Ø X (1Ø ** - 3Ø8), com precisão decimal de até 15 dígitos e utilização da notação científica para representar números com mais de 15 dígitos.

vi.Currency

Valores numéricos monetários com/sem sinal de até 13 caracteres e 16 dígitos, apresentados no formato especificado pelo usuário.

vii.Date

Datas de até 3Ø caracteres, representadas no formato especificado pelo usuário.

viii.Time

Horas de até 2Ø caracteres, representadas no formato especificado pelo usuário.

c. Dicionário de dados

Guarda informações da aplicação relativas à base de dados, programas, telas e relatórios. O usuário pode consultar, gerar referência cruzada e atualizar informações.

d. Liguagem de definição e manipulação de dados

Uma linguagem relacional SQL num padrão diferente do IBM, permite recuperar, manipular, definir e proteger dados, estruturas de tabelas e visões lógicas associadas a cada tabela de um banco de dados relacional.

Não se pode usar palavras reservadas como nomes de tabelas, campos ou variáveis. Não há posição fixa dos comandos na linha e no número de comandos por linha, podendo ser delimitados por vários brancos. Palavras reservadas podem ser abreviadas até o menor número de caracteres que permita distinguí-las, e não devem ser usadas para qualificar nomes de arquivos, campos e variáveis de memória.

Telas de auxílio ajudam o usuário na sintaxe e uso dos comandos, na codificação de programas e depuração de erros.

Pode-se usar o editor de textos RBEDIT ("RBASE EDITOR") ou outro editor com o qual o usuário esteja familiarizado. A execução dos comandos pode ser feita de modo interativo por menus, por comandos internamente ao RBASE SYSTEM V ou através da leitura de arquivos externos previamente armazenados na memória secundária.

A entrada de dados e comandos pode ser feita em massa via leitura de arquivos externos ou via tela que deve conter os campos do arquivo a serem atualizados. Cada arquivo pode estar associado à várias telas e uma tela pode realizar a entrada de dados em vários arquivos, simultaneamente.

Regras de validação podem ser criadas durante a definição do banco de dados para conferir os valores fornecidos pelo usuário durante a entrada de dados quanto a comprimento,

formato e conteúdo, podem ser até 20 por tabela e conter até 10 condições e, ficam associadas aos campos no arquivo físico e armazenadas do dicionário de dados. Assim, não precisam ser especificadas a cada tela e, entradas de dados são criticadas, automaticamente. As regras permitem a realização de consultas à várias tabelas para verificar a validade dos dados, porém, não se pode incluir comandos SQL na sua especificação. Procedimentos de validação adicionais devem ser embutidos nos programas que ativam as telas, para validar os dados antes da atualização do arquivo.

A linguagem dispõe, também, de comandos para a emissão de relatórios. A especificação de caracteres de formatação pelo usuário, adapta o relatório ao formato desejado.

Pode-se acessar dados usando-se a linguagem conversacional PBE ("Prompt-By-Example"). Que permite consultar, definir e atualizar informações em uma ou mais tabelas sem conhecer a sintaxe dos comandos. A linguagem visual e intuitiva permite buscar valores similares aos especificados na consulta, para detectar erros de datilografia e informações imprecisas fornecidas pelo usuário.

e. Interface com usuário

Uma interface interativa por menus permite desenvolver aplicações navegando-se entre telas, sem conhecimento da sintaxe dos comandos. A escolha de comandos e argumentos, gera programas que são armazenados para formar aplicações.

Interfaces permitem acessar/fornecer dados de/para arquivos tipo WKS (Lotus 123), DBF (dBase II e III), DIF (VisiCalc), PFS (IBM Professional Files), SYL (Multiplan) e ASCII.

f. Integração com outros SGBD's

As aplicações desenvolvidas nesta linguagem podem ser executadas em outros SGBS's que utilizem a linguagem SQL. Porém, as aplicações necessitam de algumas alterações para se adequarem ao padrão SQL utilizado pelo SGBD destino.

A importaçao/exportação de dados é feita pelo FGW ("File GateWay"), que permite transferir arquivos entre bancos de dados diferentes, independente do porte do ambiente usado, mas obedecendo os formatos de dados reconhecidos pelo SGBD. Durante a importação, a estrutura do arquivo origem é lida e copiada para o RBASE. O usuário pode modificá-la antes da conversão estar completa, associando o tipo de dados Rbase mais próximo daquele do arquivo origem. Assim, é possível copiar arquivos tipo WKS, PFS, SYL, DIF, DBF e ASCII. Porém, não há integração entre o RBASE e outros SGBD's.

Pode-se executar programas escritos em linguagens não SQL, internamente ao SGBD. O "ZIP" permite, assim, adicionar características de outros produtos. Porém, são válidos, apenas, programas que podem ser executados diretamente no DOS, excluindo a ativação de utilitários DOS (ex: Word). O controle do sistema operacional passa para o novo produto durante sua execução e retorna ao SGBD ao final desta.

g. Qualidade da documentação

A documentação disponível é suficiente. O suporte ao SGBD é dado por representantes nacionais. Possui mensagens, telas, manuais e cursos redigidos em inglês, dificultando a compreensão de usuários leigos neste idioma.

h. Segurança e proteção dos dados

Um mecanismo de segurança permite especificar senhas que protegem as informações a nível de usuário, banco de dados, tabela, visão e tela, e autorizações para leituras e atualizações no banco de dados. Pode-se estabelecer níveis de acesso aos dados. O controle do acesso a informações protege os dados de atualizações indevidas.

Somente usuários autorizados podem acessar os dados e somente uma senha pode estar ativa de cada vez. Telas que acessam várias tabelas protegidas e possuem senhas de segurança própria, devem ser identificadas por meio destas que substituem as senhas das tabelas, tornando-as sem efeito a nível das operações sobre as telas.

i, Deteção de Falhas e reconstrução do banco de dados

Não há um mecanismo restaurador automático para recuperar os dados, auxiliar as transações incompletas e garantir a segurança e integridade das informações em caso de falhas de software/hardware ou falta de energia. Não é utilizado

um mecanismo automatizado para gerar cópia do banco de dados antes de cada sessão. Deste modo, os arquivos podem ser danificados e perderem parte ou todas as informações. Cabe ao usuário gerar e manter cópias do banco de dados, de forma a permitir a recuperação de dados.

A linguagem dispõe de comandos que permitem gerar cópia ("BACKUP") e restaurar ("RESTORE") o banco de dados. Deve-se tirar cópia deste a cada início de sessão, para permitir a ressauração e garantir a integridade do mesmo em caso de faltas de sistema, hardware ou falta da energia. Porém, em dados modificados após a geração do "Backup" não são recuperados e as operações precisam ser refeitas.

j. Concorrência

Em ambientes multiusuário, pode-se atualizar, consultar e incluir dados, concorrentemente, em tabelas comuns. Um mecanismo de bloqueio garante compartilhamento aos dados e controle de concorrência, sem perda de integridade e consistência. Assim, durante atualizações em um arquivo, este permanece protegido e inacessível por outro usuário.

No momento do acesso aos objetos pelo usuário, estes são bloqueados, automaticamente, é aplicado o bloqueio menos limitante e mais breve de conformidade com a operação em execução, para permitir o maior acesso possível aos objetos pelos usuários. Outros usuários que desejarem atualizá-lo, receberão uma mensagem informando que este está em uso.

Há vários tipos de bloqueio no RBASE SYSTEM V para complementar o automático. São eles:

i, **Bloqueio de tabela**

Permite que vários usuários acessem o mesmo objeto ao mesmo tempo, porém, bloqueia a tabela corrente em edição durante atualizações e a desbloqueia ao término destas, tornando as modificações permanentes no banco de dados. Tabelas bloqueadas para atualizações permanecem desbloqueadas para leituras.

ii.Bloqueio de banco de dados

Permite que vários usuários acessem o mesmo banco de dados ao mesmo tempo. Porém, atualizações na sua estrutura por um usuário torna-o totalmente bloqueado contra qualquer modificação por outros usuários.

iii.Bloqueio Integral

Impade acesoos concomitantes ao objeto bloqueadc, Deste modo, garante-se uso exclusivo de tabelas.

Caso ocorram "deadlocks", tenta-se obter o objeto bloqucado para o outro usuário durante um período de tempo. Ao final desse período, se o objeto não tiver sido desbloqueado, é cancelada a tentativa de acesso. Deste modo, garante-se a finalização das operações que bloquearam o objeto.

Um mecanismo permite ao usuário verificar a possibilidade de ocorrer conflitos de atualização antes da realização da mesma. Assim, pode-se evitar acessos concorrentes.

k. Ambiente Operacional

Em ambientes monousuário que usem microcomputadores do tipo PC-XT ou PC-AT, é necessária uma configuração mínima de hardware de 512 Kb de memória RAM, 3,6 Mb em disco rígido e sistema operacional MS/DOS versão 2.0 ou posterior. Porém, em ambientes multiusuário, é necessário 640 Kb de memória RAM e sistema operacional MS/DOS versão 3.1 ou posterior.

l. Portabilidade e flexibilidade

O RBASE SYSTEM V e suas aplicações rodam em ambientes de microcomputadores PC's. Suas características não são padrões a diversos sistemas operacionais, não permitindo a migração de aplicações entre máquinas.

m. Gerador de Aplicações

Um gerador de aplicações modular onde cada módulo fica responsável por realizar determinadas tarefas, permite desenvolver aplicações, sem programação. A integração destes automatiza o processo de desenvolvimento. Todos os módulos possuem interface interativa por menus e podem ser ativados individualmente e externamente ao RBASE SYSTEM V.

CI DE ("Definition Express") é responsável pela definição das estruturas do banco da dados. Define-se as tabelas onde ficarão armazenados os dados e, regras de validação e

sesihas para os meimosi. As tabalas tão geradas iriternamcntr, e muasi isiformaçíes guardadas no dicionário de dador.

O AE ("Application Express") é responsável pela geração dos programas que compõem a aplicação, podendo ativar procedimentos previamente criados. O usuário responde perguntas interativas realizadas pelo módulo e seleciona funções e argumentos para compor os programas.

Cabe ao RBASE gerar o código real necessário para executar as tarefas selecionadas e formar a aplicação. É usado o formato padrão de entrada de dados (telas) e relatórios, a menos que o usuário defina as telas e relatórios usando os módulos FE e RE, respectivamente. O dicionário de dados vai sendo atualizado a medida que novas informações são fornecidas sobre a aplicação.

Alterações feitas, diretamente, no código gerado pelo PE, serão desativadas em caso de uso posterior do mesmo para modificar a aplicação, pois o PE age sobre a última versão da aplicação por ele gerada. Logo, alterações em aplicações geradas pelo PE devem ser feitas usando o PE, para evitar perdas de informações e inconsistências entre versões.

Pode-se construir aplicações por prototipação. Usa-se o PE para gerar versões sucessivas da aplicação que vão sendo refinadas e modificadas, eliminando erros e incorporando novas características. Usuários obtêm resultados práticos mais rapidamente e podem detectar falhas e necessidades nas fases iniciais da implementação.

O FE ("Forms Express") é responsável pela geração de telas de entrada de dados. A entrada de dados pode ser validada através da definição de regras de validação que checam os valores digitados pelo usuário e aumentam a precisão dos dados. Pode-se criar/atualizar até 5 tabelas ao mesmo tempo a partir da mesma tela, e criar menus com as opções de operação permitidas para cada tela.

O RE ("Reports Express") é responsável pela geração de relatórios de saída de dados. O usuário define o formato de apresentação dos dados de saída gerados pela aplicação.

Não pode-se gerar aplicações pelos módulos "EXPRESS" sem o uso de menus. Logo, torna-se inviável sua aplicabilidade a esta tese que fará a entrada de dados e comandos em massa via leitura de arquivos externos sem o uso de telas, para eliminar a improdutividade gasta com a seleção de opções em menus e, digitação e fornecimento de informações via tela.

n. Arquitetura

Não foi possível obter informações sobre a arquitetura do SGBD. A documentação disponível não relata detalhes de implementação e da arquitetura usada pelo RBASE SYSTEM V.

o. Utilitários

Uma versão avançada de compilador ("CodaLock") transforma arquivos de comandos RBASE em arquivos binário. Os comandos são analisados em tempo de compilação para detectar erros

e, então, convertidos em código absoluto. O código torna-se ilegível e protegido, aumentando a segurança do objeto, e o tempo de execução é reduzido pela eliminação da interpretação dos comandos antes de cada execução e pela não necessidade da presença do código fonte.

Arquivos decodificados pelo "CodeLock" sofrem restrições como a não permissão para conter blocos de comandos de tela e menu em arquivos de comandos. Blocos devem ser definidos em arquivos de procedimentos. Porém, nem todos os comandos podem ser usados em aplicação compilada por falta de suporte do "CodeLock" a estes.

A execução de aplicações compiladas de forma otimizada deve ser feita pelo RRUN. O objetivo é aumentar a velocidade de execução e reduzir a utilização da CPU e memória.

A.7 ZIM

A seguir, será apresentada uma descrição do ZIM. Informações podem ser obtidas em [Curo86, Derf88, Dick86a, Hell85, Info88b, KMCo89a, KMCo89b, Seym88c, Soua88, Zant87a, Zant87b, Zant87c, Zant87d, Zant87e, Zant87f, Zant87g, Zant87h, Zant87i, Zant87j, Zant87k, Zant89].

É um SGBD relacional desenvolvido pela Zanthe Information do Canadá e, representado e comercializado no Brasil pela RIM Informática Ltda.

a. Estrutura física e organização do espaço de armazenamento

Os dados são armazenados e manipulados de forma estruturada segundo características do modelo entidade-relacionamento, sem, entretanto, perder as características do modelo relacional. Aplicações são implementadas como uma coleção de objetos (estruturas): entidades, relacionamentos, documentos, campos, campos virtuais, telas, roles, conjuntos e variáveis. O banco de dados é visto como uma coleção de conjuntos entidade e relacionamentos entre eles.

Cada categoria de entidade constitui um conjunto entidade e é representada por tabelas bidimensionais organizadas por linhas (tupla) e colunas (atributos). Além de armazenar informações sobre as entidades (atributos), pode-se também, armazenar e manipular relacionamentos entre entidades.

Relacionamentos são associações entre entidades, e possuem o mesmo tratamento destas. Logo, relacionamentos muitos-para-muitos, são representados como tabelas bidimensionais cujas colunas constituem os atributos-chave das entidades e os atributos do relacionamento, e cada linha define um elo de ligação entre valores de dois conjuntos-entidade.

Para usar eficientemente o espaço de armazenamento e facilitar atualizações nos dados, não é permitido duplicar atributos no banco de dados. Logo, conjuntos entidade que precisam acessar informações a respeito de outras entidades, devem fazê-lo através dos relacionamentos e chaves estrangeira. Estes funcionam como elo de ligação

permanente entre entidades, devendo ser modificados somente em caso de alterações no banco de dados.

Documentos são arquivos armazenados externamente ao ZIM (inclui-se arquivos do sistema operacional que necessitam ser acessados) e dispositivos externos. Pode-se criar arquivos de comandos (programas, macros e procedimentos) e de dados, armazená-los na memória secundária e, posteriormente, acessá-los internamente ao SGBD.

Campos são itens de dados que compõem as entidades, relacionamentos e documentos. Durante a criação de um conjunto entidade/relacionamento, pode-se definir campos de índice que serão usados para acessar os dados. Estes permanecem associados a entidade, porém, pode-se deletar e criar novos campos. Não há limite na quantidade e tamanho dos índices que são gerenciados, automaticamente, pelo ZIM.

Campos virtuais são campos derivados de outros campos, cujos conteúdos não são armazenados fisicamente no banco de dados, mas recebem tratamento igual aos outros. Podem ser campos concatenados, subcampos, chaves parciais ou compostas, ou valores calculados. O campo virtual é, dinamicamente, gerado sempre que forem realizados acessos a registros, evitando a codificação de expressões de cálculo diversas vezes. Podem ser estabelecidas chaves de acesso através da indexação dos mesmos, entretanto, os valores de índice são armazenados fisicamente no banco de dados.

Telas são mapas formatados de E/S. Há cadastros de telas especificamente projetados para controlar tipo, localização e apresentação das informações. Pode-se definir formalmente

grupos de telas a serem usados para realizar E/S em múltiplos objetos ZIM.

"Roles" são usados para definir subconjuntos e sinônimos para entidades/relacionamentos. Estes ficam armazenados no banco de dados. Pode-se definir nomes abreviados como sinônimos dos objetos para agilizar a codificação das aplicações, e pode-se representar hierarquias de classes pela definição de classes e subclasses de objetos.

Conjuntos são entidades temporárias criadas, dinamicamente, dos conjuntos entidades/relacionamentos, para permitir a manipulação de partes do banco de dados. Assim, não é gerada cópia dos dados e os conjuntos são deletados ao fim de cada sessão. São criados índices que informam ao ZIM a posição dos registros na memória. Atualizações em conjuntos são, automaticamente, propagadas para os objetos origem e vice-versa, exceto para os objetos de tipo relacionamento.

Variáveis são objetos temporários criados durante uma sessão para armazenar valores de dados. O conteúdo destas é, automaticamente, perdido ao fim de cada sessão.

Dados são acessados/armazenados usando o método sequencial indexado implementado pela árvore B+ balanceada. Cada nó é uma página do disco, e armazena um número máximo de chaves para minimizar os acessos a disco para buscar um registro.

Usa-se a paginação como técnica de organização do espaço de armazenamento secundário. Pode-se selecionar o tamanho da página a ser usada. O padrão do sistema é de 1024 Bytes.

O tamanho do banco de dados é limitado pela capacidade do disco, e o tamanho dos registros pelo tamanho de uma página do disco. Cada registro pode ter desde um campo de 32 Kbytes até 16.000 campos de 1 Byte.

Não há limite para o número de bancos de dados por máquina e para o tamanho dos conjuntos entidade e relacionamentos. Porém, este tamanho não deve ultrapassar 2 GBytes. Pode-se possuir até 9.900 entidades por banco de dados.

Os bancos de dados de uma aplicação são armazenados em diretórios ZIM que são arquivos DOS. Porém, não é permitido mudar de diretório raiz internamente ao ZIM. Deve-se deixar o SGBD e voltar ao sistema operacional para realizar a troca. Logo, deve-se organizar os objetos que interagem com uma aplicação dentro de diretórios internos ao raiz.

b. Tipos de dados

Os tipos de dados suportados pelo ZIM são:

i. Alpha

Cadeia de caracteres de comprimento fixo, sem distinção do tipo de letra (maiúsculas/minúsculas), utilizando um espaço de armazenamento pré-definido.

ii. Varalpha

Cadeia de caracteres de comprimento variável, sem distinção do tipo de letra (maiúsculas/minúsculas), usando apenas o espaço de armazenamento necessário.

iii.Char

Cadeia de caracteres de comprimento fixo, com distinção do tipo de letra (maiúsculas/ minúsculas), utilizando um espaço de armazenamento pré-definido.

iv.Varchar

Cadeia de caracteres de comprimento variável, com distinção do tipo de letra (maiúsculas/ minúsculas), utilizando somente o espaço de armazenamento necessário.

v. Int

Valores inteiros com/sem sinal entre -32.768 e 32.767, ocupando espaço de armazenamento de 2 bytes.

vi.Longint

Valores inteiros com/sem sinal entre -2.147.483.648 e 2.147.483.647, com espaço de armazenamento de 4 bytes.

vii.Vastint

Valores inteiros com/sem sinal de até 15 dígitos, ocupando espaço de armazenamento de 8 bytes.

vii.Numeric

Valores numéricos com/sem sinal armazenados no formato caracter.

viii.Date

Datas no formato YYYYMMDD ocupando espaço de armazenamento de 8 bytes.

c. Dicionário de dados

Guarda informações sobre estruturas do banco de dados, entidades, relacionamentos, regras de validação de dados, arquivos documento, telas de entrada de dados, variáveis de programas escritos em ADL e outros objetos ZIM. É uma base de dados ZIM com algumas entidades previamente definidas, que se pode consultar e atualizar usando comandos ADL.

A criação de objetos ZIM deve ser precedida da definição e especificação destes no dicionário de dados. Deve-se gerar os objetos a partir das informações armazenadas.

O dicionário não é ativo. Logo, mudanças nas especificações de objetos não são propagadas automaticamente. É preciso eliminar a versão anterior dos objetos e regerá-los.

A deleção de objetos elimina dados neles armazenados e índices a eles associados. Logo, deve-se gerar cópia destas informações para recuperá-las após a regeração dos objetos.

d. Liguagem de definição e manipulação da dadas

A ZIM/ADL ("ZIM Application Development Language") permite definição e manipulação de telas de entrada de dados, importação/exportação de dados, geração de relatórios e menus, e definição de macros e procedimentos.

Pode-se usar o editor de textos do ZIM ou qualquer outro editor para criar programas. Comandos não possuem posição

fixa na linha de comando, podendo delimitá-los por vários brancos (inclusive antes do primeiro comando na linha). As palavras reservadas podem ser abreviadas pelo mínimo de caracteres que possa distingüí-las entre si, e não podem ser usadas para qualificar nomes de objetos ZIM.

A entrada de dados e comandos pode ser feita em massa via leitura de arquivos externos ou via tela que deve conter os campos do arquivo a serem atualizados. Telas podem conter informações de vários arquivos e definições de regras de validação para a conferência dos valores fornecidos pelo usuário quanto a formato, comprimento e conteúdo. Não há limite para o número de telas construídas por aplicação. Os atributos de telas e de campos que as compõem são definidos pelo ISDF ("Interactive Form Definition Facility") que é uma ferramenta interativa por menus para especificar telas.

As regras não são associadas aos campos no arquivo físico e ficam embutidas na tela na qual foram criadas. Assim, a entrada de dados em arquivos que forem feitas sem usá-las, não serão criticadas. As regras sofrem outras limitações como a não permissão da inclusão de comandos ADL na sua especificação, a não inclusão destas no dicionário de dados e a especificação destas no atributo de validação associado ao campo na tela. Procedimentos de validação adicionais devem ser embutidos nos programas que ativam as telas, para validar os dados antes de atualizar o arquivo.

Telas de auxílio ajudam o usuário na sintaxe e uso dos comandos, na codificação de programas e depuração de erros, e possuem um índirra de auxília por tópico de programação.

A emissão de relatórios pode ser feita usando-se o padrão da linguagem ou especificando-se caracteres de formatação, para obter relatórios mais sofisticados e flexíveis. Deve-se adaptar os relatórios às necessidades de apresentação.

A definição de macros e procedimentos (documentos ZIM) permite definir e grupar comandos ADL em rotinas. Porém, a implementação do ZIM sobre o sistema operacional MS/DOS, armazena cada procedimento/macro em um arquivo DOS. Logo, aplicações complexas que encadeam muitos procedimentos têm desempenho ruim e baixa velocidade de execução, pois o MS/DOS precisa abrir e fechar vários arquivos.

Aplicações podem ser desenvolvidas interativamente por tela ou pela criação de arquivos de comandos ADL externamente ao ZIM. A execução deste documento pode ser feita internamente ao SGBD por chamadas a seu nome como se fosse um comando da ADL, ou ativando-o com o ZIM no início da sessão.

e. Interface com usuário

Interfaces interativas por menus interligam o ZIM. Pode-se acessar dados do banco de dados usando programas escritos em PL/I pela interface ZIM/PLI, e pode-se importar/exportar dados entre arquivos ASCII, Lotus 123 e dBase.

f. Integração com outros SGBD's

A importação/exportação de dados entre bancos de dados é feita no formato ASCII, Lotus 123, dBase e VisiCalc. Porém, não há integração entre o ZIM e outros SGBD's.

A adaptação do ZIM aos padrões IBM é feita pela ferramenta (ZIM/ISQL) que acessa a base de dados usando comandos SQL. Aplicações desenvolvidas para outros SGBD's que usam SQL, podem acessar os dados da base de dados ZIM.

g. Qualidade da documentação

A documentação sobre o ZIM não é muito vasta. O suporte ao SGBD é dado por representantes nacionais. Seus manuais, telas, mensagens e comandos são em inglês, dificultando a compreensão de usuários leigos neste idioma.

h. Segurança e proteção dos dados

Um mecanismo de segurança protege acessos a relacionamentos /entidades, campos da base de dados e ao próprio SGBD, via definição de senhas e níveis de acesso aos dados. O usuário deve se identificar ao entrar no ZIM e somente pode acessar objetos a ele autorizados. Aplicações ficam protegidas contra alterações indevidas.

Um compilador de segurança (ZIM/CS - "Compiler Security"), gera código criptografado não alterável por usuários não autorizados e não legível externamente ao ZIM. Porém, o ZIM precisa descriptografar objetos ao acessá-los, aumentando o tempo de improdutividade e prejudicando seu desempenho.

i. Deteção de Falhas e reconstrução do banco de dados

Um mecanismo recuperador de dados JI("Journal Image") anula transações incompletas e garante a reconstrução dos dados. Para permitir a restauração destes em caso de falhas de meio (hardware ou queda de energia), é mantido um arquivo em disco que guarda as atualizações efetuadas com sucesso no banco de dados ("JOURNAL FILE"). Caso ocorram falhas, estas atualizações são reexecutadas sobre cópia do banco de dados previamente gerada, para recriar o estado corrente e consistente das informações. Para tanto, deve-se gerar cópia do mesmo antes de iniciar cada sessão.

Ao iniciar uma transação de atualização, é gerada cópia dos dados em arquivo. As atualizações são feitas neste arquivo e guardadas no banco de dados ao término normal das mesmas. Em falhas de transação, o banco de dados não é alterado e permanece consistente com a versão que iniciou a transação.

O "UNLOAD" permite gerar cópia do banco de dados em arquivo documento ZIM. O usuário deve ativá-lo antes ou depois de cada sessão, para permitir a recuperação dos dados em caso de falhas durante a mesma. A ativação do "UNLOAD" pode ser incluída no procedimento automático realizado pelo SGBD no início e fim de cada sessão. A restauração do arquivo danificado é feita pelo "RELOAD" que copia o documento gerado pelo "UNLOAD" sobre ele próprio ("baixa de backup").

j. Concorrência

Em ambientes multiusuário, dados podem ser compartilhados e transações executadas em paralelo. Entretanto, transações de atualização no banco de dados são executadas em série e, dados acessados por estas permanecem inalterados perante outros usuários até que a mesma esteja completa.

Um mecanismo de bloqueio restringe o acesso as informações compartilhadas. Dados em atualização por uma transação ficam totalmente bloqueados, não podendo ser acessados por nenhum usuário. Porém, dados em leitura são bloqueados para atualização por outras transações, podendo ser lidos.

Um mecanismo de reserva permite reservar partes do banco de dados, bloqueando-as de outros usuários durante operações de atualização. A reserva permanece até o fim da transação.

Caso ocorram "deadlocks", o ZIM aborta uma das transações e libera o arquivo. Não é possível alterar especificações de objetos e do banco de dados para evitar inconsistências.

k. Ambiente Operacional

Em ambientes monousuário que usem microcomputadores do tipo PC-XT ou PC-AT, é necessária uma configuração mínima de hardware de 512 Kb de memória RAM, 1 Mb em disco rígido e sistema operacional MS/DOS versão 2.1 ou posterior. Porém, em ambientes multiusuário, é necessário sistema operacional MS/DOS versão 3.1 ou posterior.

1. Portabilidade e flexibilidade

O ZIM e suas aplicações rodam em ambientes mono e multi-usuário, incluindo mainframes e microcomputadores. Suas características são padrões a sistemas operacionais como MS/DOS, UNIX, QNX (UNIX para PC), XENIX (UNIX para PC-AT), OS/2, EDIX (EDISA), SIDIX (SID), DIGIX (Digirede), VMS (VAX, Elebra), SOX, AOS-VS (Cobra-1000, Data General), HP-UX (HP9000/800) e VM/CMS (IBM, Labo8090, I9000).

m. Gerador de Aplicações

O Gerador de Aplicações ZIM/DA ("ZIM Development Assistant) permite definir, criar e manter o dicionário de dados, bases de dados, telas, relatórios, menus e outros objetos, sem programação. Interfaces interativas por menus dispensam o usuário do conhecimento da sintaxe dos comandos ADL.

O ZIM/DA compõe-se de componentes que geram aplicações orientadas a menu. Um deles é o ZIM/IDD ("Zim Interactive Data Dictionary") que gera e mantém o dicionário de dados.

n. Arquitetura

Não foi possível obter informações sobre a arquitetura do SGBD. A documentação disponível não relata detalhes de implementação e da arquitetura utilizada pelo PARADOX.

o. Utilitários

O compilador ZC ("Zim Compiler") traduz comandos ADL em código objeto, acelera a execução de aplicações e protege-as de tentativas de violação no código fonte. Usuários recebem apenas módulos objetos e não podem fazer leituras e modificações nos mesmos. Assim, protege-se o código fonte de acessos indevidos. Porém, o compilador restringe o escopo de comandos que podem ser usados nas aplicações.

O ZIM/RUNTIME permite a execução de aplicações compiladas, otimizando tempo de execução e, consumo de memória e CPU. O ZIM/QUERY RUNTIME possui capacidade adicional para realizar consultas e criar relatórios instantâneos, além daqueles já existentes na aplicação.

APÊNDICE B

SISTEMAS GERENCIADORES DE BANCO DE DADOS NÃO UTILIZADOS

A seguir, serão brevemente caracterizados alguns dos SGBD's analisados e referenciados na tese, e **não utilizados** devido aos fatores descritos no capítulo IV. A fim de facilitar o acesso as informações, estas serão apresentadas em ordem alfabética. Algumas características serão omitidas por não terem sido referenciadas na literatura.

ARCO-IRIS [Soua88, Tecn88, Tecn89a, Tecn89b]

é um SGBD desenvolvido, representado e comercializado no Brasil pela Tecnosoft - Tecnologia de Software Ltda.

Em ambientes monousuário que usem microcomputadores do tipo PC-XT , PC-AT, é necessária uma configuração mínima de hardware de 640 Kb de memória RAM, disco rígido e sistema operacional MS-DOS versão **2.0** ou posterior.

O SGBD dispõe de interface interativa por menus, gerenciador de telas, gerador de relatórios e gráficos de distribuição de dados, mecanismo de segurança que verifica a legitimidade de acessos ao banco de dados por usuários e Sistema Integrado de Ajuda ao Usuário. Uma interface de programação permite manipular bancos de dados ARCO-ÍRIS através das linguagens de programação C ou Pascal.

Uma linguagem permite consultar e atualizar o banco de dados e imprimir relatórios. Os comandos podem ser grupados em procedimentos que são executados em modo não interativo.

DATAEASE [Brow86b, Corr88, Derf88, Info88a, Palm87, Petr87, Poor87, Poor89, Sand88, Sant87b, Seym88c, Soua88]

é um SGBD relacional desenvolvido pela Dataease International Inc. e, representado e comercializado no Brasil pela Planconsult Tecnologia Ltda.

Em ambientes monousuário que usem microcomputadores do tipo PC-XT ou PC-AT, é necessária uma configuração mínima de hardware de 640 Kb de memória RAM, disco rígido e sistema operacional MS-DOS versão 2.0 ou posterior. Em ambientes multiusuário, o sistema operacional deve ser MS-DOS versão 3.1 ou superior.

A linguagem DQL (Dataease Query Language) é relacional. A programação da aplicação pode ser feita via seleção de comandos em menus ou leitura de arquivos externos.

Os dados são armazenados em tabelas segundo características dos modelos relacional e entidade-relacionamento. Pode-se estabelecer vários relacionamentos entre tabelas, criar até 26 bases de dados em um mesmo meio físico e acessar dados de várias tabelas ao mesmo tempo.

Dados são acessados e armazenados usando a indexação por árvore B. Porém, a manipulação de campos indexados é lenta e prejudica o desempenho do SGBD.

O DATAEASE dispõe de gerador de aplicações, relatórios, telas e cópias de segurança, mecanismo de restauração de arquivos, importador/exportador de dados de arquivos ASCII, Lotus 1-2-3, dBase e DIF, e interface por menus que permite ao usuário desenvolver aplicações sem programação.

O "DATAEASE LAN" converte aplicações monousuária em multiusuária, e o "DATAEASE CONNECT" converte aplicações em PC's para mainframe e vice-versa. Pode-se desenvolver e documentar aplicações (geração de manuais de usuário e do sistema), automaticamente, pelo "DATAEASE DEVELOPER". O "DATAEASE EXEC" otimiza a execução de aplicações.

A segurança dos dado!:; é garantida pelo mecanismo de senhas e níveis de acesso às informações. Um usuário pode somente desempenhar tarefas e acessar dados a ele autorizados.

DATAFLEX [Barb86, Chrs85, Corr88, Curo86, Derf88, Info88a, Info88e, Intr89a, Intr89b, Mars86, Petr87, Petr89, Seym88c, Wong88]

é um SGBD desenvolvido pela Data Access Corporation e, representado e comercializado no Brasil pela Intercomp - Interamericana de Computação Ltda. Apesar de comercializado como relacional, o modelo lógico deste SGBD é próprio, com características similares ao modelo hierárquico.

Em ambientes monousuário que usem microcomputadores do tipo PC-XT ou PC-AT, é necessária uma configuração mínima de hardware de 640Kb de memória RAM, 900 Mb em disco rígido e sistema operacional MS/DOS versão 3.0 ou superior.

Pode ser utilizado em ambientes mono e multiusuário UNIX, XENIX, AIX, VAX/VMS, OS/2, CP/M, MS/DOS. É possível migrar aplicações entre estes ambientes.

O SGBD dispõe de gerador de relatórios, menus, aplicações, programas de manutenção de arquivos e etiquetas, interface por menus, conversor de arquivos dBase para DATAFLEX, importador/exportador de dados e dicionário de dados.

A entrada de dados pode ser feita via tela com validação de valores. Regras de validação ficam associadas aos campos para garantir a validade dos dados fornecidos pelo usuário.

Dados são armazenados em um banco de dados compartilhado entre vários equipamentos, que não, necessariamente, usam o mesmo sistema operacional. Os arquivos são compostos de registros de tamanho fixo que são acessados e armazenados fazendo uso do método sequencial indexado implementado usando árvore-B+ como estrutura de armazenamento.

Deleções em registros liberam, automaticamente, o espaço em disco, eliminando a necessidade de compactação do banco de dados. O usuário pode pré-alocar espaço para seus arquivos, evitar a fragmentação dos mesmos e aumentar seu desempenho.

A segurança dos dados é feita por mecanismos de senhas que garantem acesso ao banco de dados por usuários autorizados.

Linguagens permitem definir/manipular arquivos, desenvolver aplicações, emitir relatórios, comunicar-se com o sistema operacional e acessar arquivos não DATAFLEX. Pode-se otimizar a linguagem original para adicionar comandos ou macros criadas pelo usuário, e alterar rotinas básicas do

produto. A documentação e telas são em português para facilitar o uso e a comunicação com o usuário.

INGRES FOR PC [Fink88, Pasc87, POMu88a, PCMu88b, Shaw88]

é um SGBD relacional desenvolvido na universidade da Califórnia e, representado e comercializado pela Relational Technology Inc.

Em ambientes monousuário que usem microcomputadores do tipo PC-XT ou PC-AT, é necessária uma configuração mínima de hardware de 640Kb de memória RAM, 10 Mb em disco rígido e sistema operacional MS/DOS versão 2.1 ou posterior.

Pode ser usado em ambientes mono e multiusuário UNIX, VMS, VM, VAX, IBM/MVS, CM/CMS, MS/DOS, e estações de trabalho SUN e APOLLO. Pode-se migrar aplicações entre eles.

O SGBD usa as linguagens relacionais SQL no padrão IBM e QUEL. Pode-se executar aplicações desenvolvidas em outros SGBD's usando SQL.Programas desenvolvidos em C podem conter comandos SQL, desde que usem o ESQL ("Embedded SQL").

A interface é interativa por menus, permitindo ao usuário consultar o banco de dados e desenvolver aplicações sem conhecer a sintaxe dos comandos. Os dados podem ser visto de forma tabular nas telas.

O usuário dispõe da facilidade QBE ("Query-By-Example"). Telas padrões para tabelas e visões podem ser criadas usando o módulo QBF ("Query-By-Forms"). Pode-se consultar e manter o banco de dados usando a QBF no modo QBE.

O SGBD dispõe de gerador de relatórios ("Report Writer") com facilidade RBF ("Report-By-Form") para projetar relatórios na tela, gerador de aplicações com facilidade ABF ("Applications-By-Forms") para desenvolver aplicações complexas num ambiente integrado, interface para acessos a arquivos DBASE, importador/exportador de dados para arquivos ASCII, e dicionário de dados. A linguagem QUEL é integrada com a ferramenta QBF, com a linguagem SQL e com o gerador de relatórios, e dispõe de um gerador de telas.

Dados são armazenados em tabelas, podendo-se estabelecer relacionamentos entre elas. Um mecanismo de alocação dinâmica de memória é usado para melhorar o desempenho do SGBD. O número de bases de dados, tabelas e registros por tabela fica limitado a disponibilidade de espaço em disco.

As técnicas de acesso/armazenamento usadas pelo INGRES são: indexação por árvore B e compressão em tabelas de grande porte, e "heap" e compressão em tabelas de pequeno porte. A criação de vistas permite que usuários diferentes vejam a mesma informação de maneira distinta. As vistas não ocupam espaço em disco e devem ser usadas para acessar tabelas, reduzindo o esforço de manutenção durante modificações na estrutura lógica da base de dados. Informações sobre bases de dados, tabelas, vistas, índices e aplicações ficam armazenadas num catálogo (dicionário de dados).

APÊNDICE C

BENCHMARKS IMPLEMENTADOS EM CADA SGBD

A seguir, serão apresentados os benchmarks de alto e baixo nível implementados em cada SGBD. Usou-se as linguagens e recursos de cada produto na codificação.

Para facilitar a compreensão dos benchmarks, os nomes dos arquivos de banco de dados e índices, e dos campos destes arquivos serão apresentados por extenso, conforme modelo E-R apresentado no capítulo V e figuras descritivas dos atributos e dados das relações apresentadas no apêndice D. A execução dos benchmarks usou nomes abreviados, para simplificar a codificação e por alguns produtos limitarem o comprimento das palavras a 8 caracteres. Adotou-se como padrão o uso dos 8 primeiros caracteres do nome.

A fim de facilitar a apresentação dos benchmarks, estes serão grupados por SGBD e, a sequência em que serão relatados seguirá a ordem de apresentação dos produtos no capítulo IV. Isto é, a sequência será estritamente alfabética, não havendo, portanto, nenhuma outra conotação na sua precedência. Em cada grupo, os benchmarks obedecerão a ordem das transações típicas Ti presentes no capítulo V.

C.1 COPPEREL

A seguir, serão apresentados os benchmarks de baixo nível implementados no COPPEREL.

a. Seleção por valor

```
ABRIR SESSAO PARA US aluno/coppe;
ABRIR BASE DE DADOS benchrel;
SELECIONAR fabricante
   TAL QUE sigla-fabricante = "elebra" EM $fabriaux;
FECHAR BASE DE DADOS;
FECHAR SESSAO;
ENC
```

b. Seleção por intervalo de valores

```
ABRIR SESSAO PARA US aluno/coppe;
ABRIR BASE DE DADOS benchrel;
SELECIONAR fabricante
   TAL QUE sigla-fabricante >= "compu"
   E sigla-fabricante <= "soft" EM $fabriaux;
FECHAR BASE DE DADOS;
FECHAR SESSAO;
ENC
```

c. Projeção

```
ABRIR SESSAO PARA US aluno/coppe;
ABRIR BASE DE DADOS benchrel;
ESTREITAR fabricante PARA sigla-fabricante EM $fabriau;
FECHAR BASE DE DADOS;
FECHAR SESSAO;
ENC
```

d. União

```
ABRIR SESSAO PARA US aluno/coppe;
ABRIR BASE DE DADOS benchrel;
UNIR fornecedor COM fabricante EM $fabrforn;
FECHAR BASE DE DADOS;
FECHAR SESSAO;
ENC
```

e. Interseção

```
ABRIR SESSAO PARA US aluno/coppe;
ABRIR BASE DE DADOS benchrel;
CRUZAR fornecedor COM fabricante EM $fabrforn;
FECHAR BASE DE DADOS;
FECHAR SESSAO;
ENC
```

f. Diferença

```
ABRIR SESSAO PARA US aluno/coppe;
ABRIR BASE DE DADOS benchrel;
SUBTRAIR fabricante DE fornecedor EM $fabrforn;
FECHAR BASE DE DADOS;
FECHAR SESSAO;
ENC
```

g. Divisão

```
ABRIR SESSAO PARA US aluno/coppe;
ABRIR BASE DE DADOS benchrel;
GRUPAR produto POR sigla-fornecedor
   TAL QUE sigla-tipo-equipamento
   CONTENHA sigla-tipo-equipamento
   DE tipo-equipamento EM $forn;
FECHAR BASE DE DADOS;
FECHAR SESSAO;
ENC
```

h, Produto Cartesiano

```
ABRIR SESSAO PARA US aluno/coppe;
ABRIR BASE DE DADOS benchrel;
CONCATENAR fabricante COM tipo-equipamento EM $prodaux;
FECHAR BASE DE DADOS;
FECHAR SESSAO;
ENC
```

i. Junção

```
ABRIR SESSAO PARA US aluno/coppe;
ABRIR BASE DE DADOS benchrel;
JUNTAR produto COM tipo-equipamento
TAL QUE sigla-tipo-equipamento= sigla-tipo-equipamento
   EM $prodaux;
FECHAR BASE DE DADOS;
FECHAR SESSAO;
ENC
```

j. Seleção + Junção

```
ABRIR SESSAO PARA US aluno/coppe;
ABRIR BASE DE DADOS benchrel;
JUNTAR aquisição COM configuração-aquisição
   TAL QUE num-aquisição = num-aquisição EM $temp1;
SELECIONAR temp1
   TAL QUE sigla-orgão-solicitante = "deoi"
   EM $aquideoi;
FECHAR BASE DE DADOS;
FECHAR SESSAO;
ENC
```

k. Subtração + União

```
ABRIR SESSAO PARA US aluno/coppe;
ABRIR BASE DE DADOS benchrel;
MODIFICAR produto TAL QUE: data-última-compra > 0
   (data-última-compra
    PARA data-última-compra - 10000);
FECHAR BASE DE DADOS;
FECHAR SESSAO;
ENC
```

l. Seleção + Subtração + União

```
ABRIR SESSAO PARA US aluno/coppe;
ABRIR BASE DE DADOS benchrel;
MODIFICAR fornecedor
```

```
      TAL QUE: sigla-fornecedor = "veiga som"
      (nome-fornecedor PARA "veiga som eletronica ltda");
   FECHAR BASE DE DADOS;
   FECHAR SESSAO;
   ENC
```

m. Seleção + Projeção

```
   ABRIR SESSAO PARA US aluno/coppe;
   ABRIR BASE DE DADOS benchrel;
   SELECIONAR produto
      TAL QUE sigla-fornecedor = "compumicro" EM $prod1;
   ESTREITAR prod1
      PARA sigla-tipo-equipamento EM $prodaux;
   FECHAR BASE DE DADOS;
   FECHAR SESSAO;
   ENC
```

n. Seleção + Subtração

```
   ABRIR SESSAO PARA US aluno/coppe;
   ABRIR BASE DE DADOS benchrel;
   RETIRAR aquisição
      TAL QUE sigla-orgão-solicitante = "deec";
   FECHAR BASE DE DADOS;
   FECHAR SESSAO;
   ENC
```

o. Seleção + Projeção + Junção

```
   ABRIR SESSAO PARA US aluno/coppe;
   ABRIR BASE DE DADOS benchrel;
   JUNTAR aquisição COM configuração-aquisição
      TAL QUE num-aquisição = num-aquisição EM $temp1;
   SELECIONAR temp1
      TAL QUE sigla-orgão-solicitante = "dere"
      EM $temp2;
   ESTREITAR temp2 PARA sigla-tipo-equipamento
      EM $aquidere;
   FECHAR BASE DE DADOS;
   FECHAR SESSAO;
   ENC
```

p. Seleção + União

```
   ABRIR SESSAO PARA US aluno/coppe;
   ABRIR BASE DE DADOS benchrel;
   SELECIONAR fornecedor
      TAL QUE sigla-fornecedor >= "micro" EM $temp1;
   SELECIONAR fabricante
      TAL QUE sigla-fabricante >= "micro" EM $temp2;
   UNIR temp1 COM temp2 EM $fabrforn;
   FECHAR BASE DE DADOS;
   FECHAR SESSAO;
   ENC
```

q. Seleção + Projeção + Junção + Subtração + União

```
   ABRIR SESSAO PARA US aluno/coppe;
```

```
ABRIR BASE DE DADOS benchrel;
JUNTAR aquisição COM configuração-aquisição
   TAL QUE num-aquisição = num-aquisição EM $temp1;
SELECIONAR temp1
   TAL QUE sigla-orgão-solicitante = "deoi"
       E num-memo-solicitação > 106500 EM $temp2;
ESTREITAR temp2
   PARA num-aquisição, sigla-tipo-equipamento
       quantidade-configuração EM $aquideoi;
MODIFICAR aquideoi TAL QUE: num-aquisição > 0
   (quantidade-configuração PARA
    quantidade-configuração + 1);
FECHAR BASE DE DADOS;
FECHAR SESSAO;
ENC
```

r. Seleção + Projeção + Junção + Subtração

```
ABRIR SESSAO PARA US aluno/coppe;
ABRIR BASE DE DADOS benchrel;
JUNTAR produto COM fabricante
   TAL QUE sigla-fabricante = sigla-fabricante
   EM $temp1;
SELECIONAR temp1
   TAL QUE sigla-tipo-equipamento = "mc01" EM $temp2;
ESTREITAR temp2
   PARA sigla-fabricante, nome-fabricante EM $temp3;
SUBTRAIR temp3 DE fabricante EM $fabriaux;
FECHAR BASE DE DADOS;
FECHAR SESSAO;
ENC
```

s. Seleção + Projeção + Junção + União

```
ABRIR SESSAO PARA US aluno/coppe;
ABRIR BASE DE DADOS benchrel;
JUNTAR produto COM fabricante
   TAL QUE sigla-fabricante = sigla-fabricante
   EM $temp1;
SELECIONAR temp1
   TAL QUE sigla-tipo-equipamento = "mc01" EM $temp2;
ESTREITAR temp2
   PARA sigla-fabricante, nome-fabricante EM $temp3;
UNIR temp3 COM fornecedor EM $fornaux;
FECHAR BASE DE DADOS;
FECHAR SESSAO;
ENC
```

C.2 DBASE III/PLUS

A seguir, serão apresentados os benchmarks de baixo nível implementados no DBASE III/PLUS.

a. Seleção por valor

```
USE fabricante INDEX sigla-fabricante
SEEK "elebra"
QUIT
```

b. Seleção por Intervalo de valores

```
USE fabricante INDEX sigla-fabricante
COPY TO fabriaux FOR sigla-fabricante >= "compu"
   .AND. sigla-fabricante <= "soft"
QUIT
```

c. Projeção

```
USE fabricante INDEX sigla-fabricante
COPY TO fabriaux FIELDS sigla-fabricante
   FOR sigla-fabricante > " "
QUIT
```

d. União

```
USE fabricante INDEX sigla-fabricante
APPEND FROM fornecedor
QUIT
```

e. Interseção

```
SELECT 2
USE fornecedor ALIAS aa
SELECT 1
USE fabricante
JOIN WITH aa TO fabrforn
   FOR sigla-fabricante= aa->sigla-fornecedor
   FIELDS sigla-fabricante, nome-fabricante
QUIT
```

f. Diferença

A linguagem não possui recursos que permitam implementar a operação relacional de diferença. Informações adicionais podem ser obtidas no capítulo V.

g. Divisão

A linguagem não possui recursos que permitam implementar a operação relacional de divisão. Informações adicionais podem ser obtidas no capítulo V.

h. Produto Cartesiano

```
SELECT 2
USE tipo-equipamento ALIAS aa
SELECT 1
USE fabricante
JOIN WITH aa TO prodaux FOR sigla-fabricante > " "
QUIT
```

i. Junção

```
SELECT 2
```

```
USE tipo-equipamento ALIAS aa
SELECT 1
USE produto
JOIN WITH aa TO prodaux FOR sigla-tipo-equipamento
   = aa -> sigla-tipo-equipamento
QUIT
```

j. Seleção + Junção

```
SELECT 2
USE configuração-aquisição ALIAS aa
SELECT 1
USE aquisição
JOIN WITH aa TO aquideoi
   FOR num-aquisição = aa -> num-aquisição
   .AND. sigla-orgão-solicitante = "deoi"
QUIT
```

k. Subtração + União

```
USE produto
REPLACE ALL data-última-compra
   WITH data-última-compra - 10000
QUIT
```

l. Seleção + Subtração + União

```
USE fornecedor INDEX sigla-fornecedor
REPLACE nome-fornecedor
   WITH "veiga som eletronicos ltda"
   FOR sigla-fornecedor = "veiga som"
QUIT
```

m. Seleção + Projeção

```
USE produto INDEX prodforn
COPY TO prodaux FIELDS sigla-tipo-equipamento
   FOR sigla-fornecedor = "compumicro"
QUIT
```

n. Seleção + Subtração

```
USE aquisição INDEX num-aquisição1
DELETE FOR sigla-orgão-solicitante = "deec"
QUIT
```

c). **Seleção + Projeção + Junção**

```
SELECT 2
USE configuração-aquisição ALIAS aa
SELECT 1
USE aquisição
JOIN WITH aa TO aquidere
   FOR num-aquisição = aa -> num-aquisição
   .AND. sigla-orgão-solicitante = "dere"
   FIELDS aa -> sigla-tipo-equipamento
QUIT
```

p. Seleção + União

```
USE fabricante INDEX sigla-fabricante
COPY TO fabrforn FOR sigla-fabricante >= "micro"
USE fabrforn
APPEND FROM fornecedor FOR sigla-fornecedor >= "micro"
QUIT
```

q. Seleção + Projeção + Junção + Subtração + União

```
SELECT 2
USE configuração-aquisição ALIAS aa
SELECT 1
USE aquisição
JOIN WITH aa TO aquideoi
   FOR num-aquisição = aa -> num-aquisição
   .AND. sigla-orgão-solicitante = "deoi"
   .AND. num-memo-solicitação > 106500
   FIELDS aa -> num-aquisição,
          aa -> sigla-tipo-equipamento,
          aa -> quantidade-configuração
USE aquideoi
REPLACE ALL quantidade-configuração
   WITH quantidade-configuração + 1
QUIT
```

r. Seleção + Projeção + Junção + Subtração

A linguagem não possui recursos que permitam implementar a operação relacional de diferença. Logo, não consegue-se implementar esta transação (T17). Informações adicionais podem ser obtidas no capítulo V.

s. Seleção + Projeção + Junção + União

```
SELECT 2
USE fabricante ALIAS aa
SELECT 1
USE produto
JOIN WITH aa TO fornaux
   FOR sigla-fabricante=aa->sigla-fabricante
   .AND. sigla-tipo-equipamento = "mc01"
   FIELDS aa -> sigla-fabricante,
          aa -> nome-fabricante
USE fornaux
APPEND FROM fornecedor
QUIT
```

C.3 DIALOG PLUS/C

A seguir, serão apresentados os benchmarks de baixo nível implementados no DIALOG PLUS/C.

a. Seleção par valor

```
USE fabricante INDICE sigla-fabricante
```

```
PESQUISE "elebra"
SAIA
```

b. Seleção por intervalo de valores

```
USE fabricante INDICE sigla-fabricante
COPIE PARA fabriaux
   QUANDO (sigla-fabricante >= "compu")
   .E. (sigla-fabricante <= "soft")
SAIA
```

c. Projeção

```
USE fabricante INDICE sigla-fabricante
COPIE PARA fabriaux CAMPOS sigla-fabricante
   QUANDO sigla-fabricante > " "
SAIA
```

d. União

```
USE fabricante INDICE sigla-fabricante
ANEXE DE fornecedor
SAIA
```

e. Interseção

```
SELECIONE 2
USE fornecedor AREA aa
SELECIONE 1
USE fabricante
JUNTE COM aa EM fabrforn
   QUANDO sigla-fabricante= aa->sigla-fornecedor
   CAMPOS sigla-fabricante, nome-fabricante
SAIA
```

f. Diferença

A linguagem não possui recursos que permitam implementar a operação relacional de diferença. Informações adicionais podem ser obtidas no capítulo V.

g. Divisão

A linguagem não possui recursos que permitam implementar a operação relacional de divisão. Informações adicionais podem ser obtidas no capítulo V.

h. Produto Cartesiano

```
SELECIONE a
USE tipo-equipamento AREA aa
SELECIONE b
USE fabricante
JUNTE COM aa EM prodaux QUANDO sigla-fabricante > " "
SAIA
```

i. Junção

```
SELECIONE a
```

```
USE tipo-equipamento AREA aa
SELECIONE b
USE produto
JUNTE COM aa EM prodaux
QUANDO sigla-tipo-equipamento=aa.sigla-tipo-equipamento
SAIA
```

j. Seleção + Junção

```
SELECIONE a
USE configuração-aquisição AREA aa
SELECIONE b
USE aquisição
JUNTE COM aa EM aquideoi
   QUANDO num-aquisição = aa.num-aquisição
   .E. sigla-orgão-solicitante = "deoi"
SAIA
```

k. Subtração + União

```
USE produto
SUBSTITUA data-última-compra
   COM data-última-compra - 10000
SAIA
```

l. Seleção + Subtração + União

```
USE fornecedor INDICE sigla-fornecedor
SUBSTITUA nome-fornecedor
   COM "veiga som eletronicos ltda"
   QUANDO sigla-fornecedor = "veiga som"
SAIA
```

m. Seleção + Projeção

```
USE produto INDICE prodforn
COPIE PARA prodaux CAMPOS sigla-tipo-equipamento
   QUANDO sigla-fornecedor = "compumicro"
SAIA
```

n. Seleção + Subtração

```
USE aquisição INDICE num-aquisiçãoi
ELIMINE QUANDO sigla-orgão-solicitante = "deec"
SAIA
```

o. Seleção + Projeção + Junção

```
SELECIONE a
USE configuração-aquisição AREA aa
SELECIONE b
USE aquisição
JUNTE COM aa EM aquidere
   QUANDO num-aquisição = aa.num-aquisição
   .E. sigla-orgão-solicitante = "dere"
   CAMPOS aa.sigla-tipo-equipamento
SAIA
```

p. Seleção + União

```
USE fabricante INDICE sigla-fabricante
COPIE PARA fabrforn QUANDO sigla-fabricante >= "micro"
USE fabrforn
ANEXE DE fornecedor QUANDO sigla-fornecedor >= "micro"
SAIA
```

q. Seleção + Projeção + Junção + Subtração + União

```
SELECIONE a
USE configuração-aquisição AREA aa
SELECIONE b
USE aquisição
JUNTE COM aa EM aquideoi
   QUANDO num-aquisição = aa.num-aquisição
   .E. sigla-orgão-solicitante = "deoi"
   .E. num-memo-solicitação > 106500
   CAMPOS aa.num-aquisição,
          aa.sigla-tipo-equipamento,
          aa.quantidade-configuração
USE aquideoi
SUBSTITUA quantidade-configuração
   COM quantidade-configuração + 1
SAIA
```

r. Seleção + Projeção + Junção + Subtração

A linguagem não possui recursos que permitam implementar a operação relacional de diferença. Logo, não consegue-se implementar esta transação (T17). Informações adicionais podem ser obtidas no capítulo V.

s. Seleção + Projeção + Junção + União

```
SELECIONE a
USE fabricante AREA aa
SELECIONE b
USE produto
JUNTE COM aa EM temp1
   QUANDO sigla-fabricante = aa.sigla-fabricante
   .E. sigla-tipo-equipamento = "mc1"
   CAMPOS aa.sigla-fabricante, aa.nome-fabricante
USE temp1
ANEXE DE fornecedor QUANDO sigla-fabricante > " "
SAIA
```

C.4 ORACLE

A seguir, serão apresentados os benchmarks de baixo nível implementados no ORACLE.

a. Seleção por Valor

```
SELECT * FROM fabricante
WHERE sigla-fabricante = 'elebra';
```

```
EXIT;
```

b. Seleção par intervalo de valores

```
SELECT * FROM fabricante
WHERE sigla-fabricante BETWEEN 'compu' AND 'soft';
EXIT;
```

c. Projeção

```
SELECT sigla-fabricante FROM fabricante;
EXIT;
```

d. União

```
SELECT * FROM fornecedor
UNION
SELECT * FROM fabricante;
EXIT;
```

e. Interseção

```
SELECT * FROM fornecedor
INTERSECT
SELECT * FROM fabricante;
EXIT;
```

f. Diferença

```
SELECT * FROM fornecedor
MINUS
SELECT * FROM fabricante;
EXIT;
```

g. Divisão

```
SELECT sigla-fornecedor FROM produto
GROUP BY sigla-fornecedor HAVING COUNT(*) = 34;
EXIT;
```

h. Produto Cartesiano

```
SELECT * FROM fabricante, tipo-equipamento;
EXIT;
```

i. Junção

```
SELECT * FROM produto, tipo-equipamento
WHERE produto.sigla-tipo-equipamento =
    tipo-equipamento.sigla-tipo-equipamento;
EXIT;
```

j. Seleção + Junção

```
SELECT * FROM aquisição, configuração-aquisição
WHERE aquisição.sigla-orgão-solicitante = 'DEOI'
AND configuração-aquisição.num-aquisição
    = aquisição.num-aquisição;
EXIT;
```

k. Subtração + União

```
UPDATE produto
SET data-última-compra = data-última-compra - 10000;
EXIT;
```

l. Seleção + Subtração + União

```
UPDATE fornecedor
SET nome-fornecedor = 'veiga som eletronicos ltda'
WHERE sigla-fornecedor = 'veiga som';
EXIT;
```

m. Seleção + Projeção

```
SELECT sigla-tipo-equipamento FROM produto
WHERE sigla-fornecedor = 'compumicro';
EXIT;
```

n. Seleção + Subtração

```
DELETE FROM aquisição
WHERE sigla-orgão-solicitante = 'DEEC';
EXIT;
```

o. Seleção + Projeção + Junção

```
SELECT configuração-aquisição.sigla-tipo-equipamento
FROM aquisição, configuração-aquisição
WHERE aquisição.sigla-orgão-solicitante = 'DERE'
AND configuração-aquisição.num-aquisição
    = aquisição.num-aquisição;
EXIT;
```

p. Seleção + União

```
SELECT * FROM fornecedor
WHERE sigla-fornecedor >= 'micro'
UNION
SELECT * FROM fabricante
WHERE sigla-fabricante >= 'micro';
EXIT;
```

q. Seleção + Projeção + Junção + Subtração + União

```
UPDATE configuração-aquisição
SET quantidade-configuração=quantidade-configuração + 1
WHERE num-aquisição
IN (SELECT num-aquisição FROM aquisição
    WHERE sigla-orgão-solicitante = 'deoi'
    AND num-memo-solicitação > 106500);
EXIT;
```

r. Seleção + Projeção + Junção + Subtração

```
SELECT * FROM fabricante
MINUS
(SELECT sigla-fabricante, nome-fabricante
 FROM produto, fabricante
```

```
 WHERE produto.sigla-tipo-equipamento = 'mc01'
 AND fabricante.sigla-fabricante =
     produto.sigla-fabricante);
EXIT;
```

s. Seleção + Projeção + Junção + União

```
SELECT sigla-fabricante, nome-fabricante
FROM produto, fabricante
WHERE produto.sigla-tipo-equipamento = 'mc01'
AND fabricante.sigla-fabricante =
    produto.sigla-fabricante
UNION
SELECT * FROM fornecedor;
EXIT;
```

C.5 PARADOX

A seguir, serão apresentados os benchmarks de baixo nível implementados no PARADOX.

a. Seleção pau valer

```
VIEW "fabricante"
MOVETO FIELD "sigla-fabricante"
LOCATE "elebra"
EXIT
```

b. Seleção por intervalo de valores

```
VIEW "fabricante"
MOVETO FIELD "sigla-fabricante"
LOCATE "compu"
teste = true
WHILE TESTE
   IF [sigla-fabricante] <= "soft"
      THEN DOWN
      ELSE teste = false
           QUITLOOP
   ENDIF
ENDWHILE
EXIT
```

c. Projeção

A linguagem PAL não possui recursos que permitam implementar a operação relacional de projeção. Informações adicionais podem ser obtidas no capítulo V.

d. União

```
VIEW "fornecedor"
ADD "fabricante" "fornecedor"
EXIT
```

e. Interseção

A linguagem PAL não possui recursos que permitam implementar a operação relacional de interseção. Informações adicionais podem ser obtidas no capítulo V.

f. Diferença

```
VIEW "fornecedor"
SUBTRACT "fabricante" "fornecedor"
EXIT
```

g. Divisão

A linguagem PAL não possui recursos que permitam implementar a operação relacional de divisão. Informações adicionais podem ser obtidas no capítulo V.

h. Produto Cartesiano

A linguagem PAL não possui recursos que permitam implementar a operação relacional de produto cartesiano. Informações adicionais podem ser obtidas no capítulo V.

i. Junção

A linguagem PAL não possui recursos que permitam implementar a operação relacional de junção. Informações adicionais podem ser obtidas no capítulo V.

j. Seleção + Junção

A linguagem PAL não dispõe de recursos que permitam implementar diretamente a operação relacional de junção, tornando inviável a implementação desta transação (T9). Informações adicionais podem ser obtidas no capítulo V.

k. Subtração + União

Esta transação representa a operação de atualização geral de arquivos (UPDATE ALL). A linguagem PAL não dispõe de comandos para atualizar campos de arquivo por batch. Isto deve ser feito no modo interativo, saindo dos objetivos fixados nesta tese. Informações adicionais podem ser obtidas no capítulo V.

l. Seleção + Subtração + União

Esta transação representa a operação de atualização parcial de arquivos (UPDATE ... WHERE). A linguagem PAL não dispõe de comandos para atualizar campos de arquivo por batch. Isto deve ser feito no modo interativo, saindo dos objetivos fixados nesta tese. Informações adicionais podem ser obtidas no capítulo V.

m. Seleção + Projeção

A linguagem PAL não dispõe de recursos para implementar diretamente a operação relacional de projeção, tornando

inviável a implementação desta transação (T12). Informações adicionais podem ser obtidas no capítulo V.

n. Seleção + Subtração

```
VIEW "aquisição"
EDIT "aquisição"
MOVETO FIELD "sigla-orgão-solicitante"
LOCATE "deec"
WHILE RETVAL
   DEL
   LOCATE NEXT "deec"
ENDWHILE
DO_IT
EXIT
```

o. Seleção + Projeção + Junção

A linguagem PAL não dispõe de recursos para implementar diretamente as operações relacionais de projeção e junção, tornando inviável implementar esta transação (T14). Informações podem ser obtidas no capítulo V.

p. Seleção + União

```
VIEW "fornecedor"
ADD  "fabricante" "fornecedor"
MOVETO FIELD "sigla-fornecedor"
LOCATE "micro"
WHILE not EOT()
      DOWN
ENDWHILE
EXIT
```

q. Seleção + Projeção + Junção + Subtração + União

Devido ao fato da linguagem PAL não dispor de recursos que permitam implementar diretamente as operações relacionais de projeção e junção, torna-se inviável implementar esta transação (T16). Informações adicionais podem ser obtidas no capítulo V.

r. Seleção + Projeção + Junção + Subtração

Devido ao fato da linguagem PAL não dispor de recursos que permitam implementar diretamente as operações relacionais de projeção e junção, torna-se inviável implementar esta transação (T17). Informações adicionais podem ser obtidas no capítulo V.

s. Seleção + Projeção + Junção + União

Devido ao fato da linguagem PAL não dispor de recursos que permitam implementar diretamente as operações relacionais de projeção e junção, torna-se inviável implementar esta transação (T18). Informações adicionais podem ser obtidas no capítulo V.

C.6 RBASE SYSTEM V

A seguir, serão apresentados os benchmarks de baixo nível implementados nu RBASE SYSTEM V.

a. Seleção por Valor

```
OPEN bd
SELECT ALL FROM fabricante
  WHERE sigla-fabricante EQ elebra
EXIT
```

b. Seleção por intervalo de valores

```
OPEN bd
SELECT ALL FROM fabricante
  WHERE sigla-fabricante GE compu
  AND sigla-fabricante LE soft
EXIT
```

c. Projeção

```
OPEN bd
PROJECT fabriaux FROM fabricante USING sigla-fabricante
EXIT
```

d. União

```
OPEN bd
UNION fornecedor WITH fabricante FORMING fabrforn
EXIT
```

e. Interseção

```
OPEN bd
INTERSECT fornecedor WITH fabricante FORMING fabrforn
EXIT
```

f. Diferença

```
OPEN bd
SUBTRACT fabricante FROM fornecedor FORMING fabrforn
EXIT
```

g. Divisão

A linguagem não possui recursos que permitam implementar a operação relacional de divisão. Informações adicionais podem ser obtidas no capítulo V.

h. Produto Cartesiano

```
OPEN bd
JOIN fabricante USING nome
  WITH tipo-equipamento USING descrição
  FORMING prodaux WHERE NE
```

```
EXIT
```

i. Junção

```
OPEN bd
JOIN produto USING sigla-tipo-equipamento
  WITH tipo-equipamento USING sigla-tipo-equipamento
  FORMING prodaux WHERE EQ
EXIT
```

j. Seleção + Junção

```
OPEN bd
JOIN aquisição USING num-aquisição
  WITH configuração-aquisição USING num-aquisição
  FORMING temp1 WHERE EQ
SELECT ALL FROM temp1
  WHERE sigla-orgão-solicitante EQ deoi
EXIT
```

k. Subtração + União

```
OPEN bd
CHANGE data-última-compra
  TO (data-última-compra - 10000)
  IN produto WHERE data-última-compra GT 0
EXIT
```

l. Seleção + Subtração + União

```
OPEN bd
CHANGE nome TO veiga som eletronicos ltda
  IN fornecedor WHERE sigla EQ veiga som
EXIT
```

m. Seleção + Projeção

```
OPEN bd
SELECT sigla-tipo-equipamento FROM produto
  WHERE sigla-fornecedor EQ compumicro
EXIT
```

n. Seleção + Subtração

```
OPEN bd
DELETE ROWS FROM aquisição
  WHERE sigla-orgão-solicitante EQ DEEC;
EXIT
```

o. Seleção + Projeção + Junção

```
OPEN bd
JOIN aquisição USING num-aquisição
  WITH configuração-aquisição USING num-aquisição
  FORMING temp1 WHERE EQ
PROJECT aquidere
  FROM temp1 USING sigla-tipo-equipamento
  WHERE sigla-orgão-solicitante EQ DERE
EXIT
```

p. Seleção + Unifa

```
OPEN bd
UNION fornecedor WITH fabricante FORMING fabrforn
SELECT sigla nome FROM fabrforn WHERE sigla GE micro
EXIT
```

q. Seleção + Projeção + Junção + Subtração + União

```
OPEN bd
JOIN aquisição USING num-aquisição
  WITH configuração-aquisição USING num-aquisição
  FORMING temp1 WHERE EQ
PROJECT aquideoi FROM temp1
  USING num-aquisição sigla-tipo-equipamento
        quantidade-configuração
  WHERE sigla-orgão-solicitante EQ deoi AND
        num-memo-solicitação GT 106500
CHANGE quantidade-configuração
  TO (quantidade-configuração + 1)
  IN aquideoi WHERE num-aquisição GT 0
EXIT
```

r. Seleção + Projeção + Junção + Subtração

```
OPEN bd
JOIN produto USING sigla-fabricante
  WITH fabricante USING sigla FORMING temp1 WHERE EQ
PROJECT temp2 FROM temp1 USING sigla nome
  WHERE sigla-tipo-equipamento EQ mc01
SUBTRACT temp2 FROM fabricante FORMING fabrforn
EXIT
```

s. Seleção + Projeção + Junção + União

```
OPEN bd
JOIN produto USING sigla-fabricante
  WITH fabricante USING sigla FORMING temp1 WHERE EQ
PROJECT temp2 FROM temp1 USING sigla nome
  WHERE sigla-tipo-equipamento EQ mc01
UNION temp2 WITH fornecedor FORMING fabrforn
EXIT
```

C.7 ZIM

A seguir, serão apresentados os benchmarks de baixo nível implementados no ZIM.

a. Seleção por Valor

```
FIND fabricante \
WHERE sigla-fabricante = elebra -> fabriaux
BYE
```

b. Seleção par intervalo de valores

```
FIND fabricante \
WHERE sigla-fabricante >= compu AND \
      sigla-fabricante <= soft -> fabriaux
BYE
```

c. Projeção

A linguagem ADL não possui recursos que permitam implementar a operação relacional de projeção, conforme a definição de banco de dados presente em [date87]. Informações adicionais podem ser obtidas no capítulo V.

d. União

A linguagem ADL não possui recursos que permitam implementar a operação relacional de união, conforme a definição de banco de dados presente em [date87]. Informações adicionais podem ser obtidas no capítulo V. Entretanto, decidiu-se implementar a operação fazendo uso dos objetivos e recursos da ADL, somente, para obtenção simulada dos parâmetros dinâmicos. O código fonte será apresentado a seguir.

```
FIND all fabricante -> temp1
FIND all fabricante -> temp2
FIND temp1 UNION temp2 -> fabrforn
BYE
```

e. Interseção

A linguagem ADL não possui recursos que permitam implementar a operação relacional de interseção, conforme a definição de banco de dados presente em [date87]. Informações adicionais podem ser obtidas no capítulo V. Entretanto, decidiu-se implementar a operação fazendo uso dos objetivos e recursos da ADL, somente, para obtenção simulada dos parâmetros dinâmicos. O código fonte será apresentado a seguir.

```
FIND all fabricante -> temp1
FIND all fabricante -> temp2
FIND temp1 INTERSECT temp2 -> fabrforn
BYE
```

f. Diferença

A linguagem ADL não possui recursos que permitam implementar a operação relacional de diferença, conforme a definição de banco de dados presente em [date87]. Informações adicionais podem ser obtidas no capítulo V. Entretanto, decidiu-se implementar a operação fazendo uso dos objetivos e recursos da ADL, somente, para obtenção simulada dos parâmetros dinâmicos. O código fonte será apresentado a seguir.

```
FIND all fabricante -> temp1
FIND all fabricante -> temp2
```

```
FIND temp1 MINUS temp2 -> fabrforn
BYE
```

g. Divisão

A linguagem ADL não possui recursos que permitam implementar a operação relacional de divisão. Informações adicionais podem ser obtidas no capítulo V.

h. Produto Cartesiano

A linguagem ADL não possui recursos que permitam implementar a operação relacional de produto cartesiano. Informações adicionais podem ser obtidas no capítulo V.

i. Junção

```
ADD RELATIONSHIPS \
     LET RELNAME = rel1 RELCONDITION = \
      'produto.sigla-tipo-equipamento = \
      tipo-equipamento.sigla-tipo-equipamento'
CREATE RELATIONSHIP rel1
FIND ALL produto rel1 tipo-equipamento -> prodaux
BYE
```

j. Seleção + Junção

```
ADD RELATIONSHIPS \
     LET RELNAME = rel2 RELCONDITION = \
       'aquisição.num-aquisição = \
       configuração-aquisição.num-aquisição'
CREATE RELATIONSHIP rel2
FIND ALL aquisição rel2 configuração-aquisição \
WHERE aquisição.sigla-orgão-solicitante=deoi->aquideoi
BYE
```

k. Subtração + União

```
CHANGE ALL produto \
LET data-última-compra = data-última-compra - 10000
BYE
```

l. Seleção + Subtração + União

```
CHANGE ALL fornecedor \
WHERE sigla-fornecedor = 'veiga som' \
LET nome-fornecedor = 'veiga som eletronicos ltda'
BYE
```

m. Seleção + Projeção

A linguagem ADL não possui recursos que permitam implementar a operação relacional de projeção. Informações adicionais podem ser obtidas no capítulo V.

n. Seleção + Subtração

```
DELETE ALL aquisição \
WHERE sigla-orgão-solicitante = 'deec'
```

```
BYE
```

o. Seleção + Projeção + Junção

```
ADD RELATIONSHIPS \
    LET RELNAME = rel2 RELCONDITION = \
      'aquisição.num-aquisição = \
      configuração-aquisição.num-aquisição'
CREATE RELATIONSHIP rel2
FIND ALL aquisição rel2 configuração-aquisição \
WHERE aquisicao.sigla-orgão-solicitante = 'dere' \
KEEP configuração-aquisição -> aquidere
BYE
```

p. Seleção + União

```
FIND ALL fabricante \
WHERE sigla-fabricante >= 'micro' -> temp1
FIND ALL fabricante \
WHERE sigla-fabricante <= bk -> temp2
FIND ALL temp1 UNION temp2 -> fabrforn
BYE
```

q. Seleção + Projeção + Junção + Subtração + União

```
ADD RELATIONSHIPS \
    LET RELNAME = rel2 RELCONDITION = \
      'aquisição.num-aquisição = \
      configuração-aquisição.num-aquisição'
CREATE RELATIONSHIP rel2
FIND ALL aquisição rel2 configuração-aquisição \
WHERE aquisição.sigla-orgão-solicitante = 'deoi'
AND aquisição.num-memo-solicitação > 106500 \
KEEP configuração-aquisição -> aquideoi
CHANGE ALL aquideoi
LET quantidade-configuração=quantidade-configuração + 1
BYE
```

r. Seleção + Projeção + Junção + Subtração

A linguagem ADL não possui recursos que permitam implementar a operação relacional de projeção. Informações adicionais podem ser obtidas no capítulo V. Adicionalmente, não é possível executar a operação de subtração que exige uma única entidade como operando, juntamente com a operação de projeção e junção que exigem pelo menos dois.

s. Seleção + Projeção + Junção + União

A linguagem ADL não possui recursos que permitam implementar a operação relacional de projeção. Informações adicionais podem ser obtidas no capítulo V. Adicionalmente, não é possível executar a operação de união que exige uma única entidade como operando, juntamente com a operação de projeção e junção que exigem pelo menos dois.

APÊNDICE D

APLICAÇÃO AEI

A aplicação AEI utilizada pelo benchmark para avaliar o desempenho dos SGBD's baseou-se no mecanismo empregado pela Eletrobrás para acompanhar a aquisição de equipamentos de informática pela empresa. No capítulo V, foi apresentado o modelo de dados da aplicação sob a forma de diagrama E-R.

A seguir, serão apresentados os objetos da AEI e suas descrições. Assim, pode-se analisar arquivos físicos, relações, atributos e chaves que compõem sua base de dados. Seus dados não serão mostrados pois constituem informações sigilosas e particulares da Eletrobrás.

O banco de dados consiste de 10 relações. Cada relação é composta de atributos. Os atributos componentes das tuplas das relações serão descritos abaixo nas figuras D.1 a D.10. Foram estabelecidas as seguintes convenções para o atributo TIPO de cada relação: A=Alfanumérico, D=Data, N=Numérico, I=Inteiro, L=lógico, T=Texto, H=Hora, M=Múltiplo (vetor).

As relações aquisição, configuração-aquisição, despacho-aquisição, evento, fabricante, fornecedor, orgão, prédio, produto e tipo-equipamento são compostas de 169, 169, 1690, 08, 33, 117, 169, 23, 23 e 34 registros, respectivamente. Os valores estão distribuídos pelas tuplas da forma como ocorrem na AEI. Não houve uniformidade na distribuição dos dados pelos registros. As tuplas de cada relação possuem tamanho fixo, porém, o tamanho varia entre relações.

NOME-ATRIBUTO	TIPO	COMPRIMENTO	COMENTÁRIO
num-aquisição	N	5	índice primário
sigla-orgão-solicitante	A	4	índice secundário
num-memo-solicitação	N	6	índice secundário
prédio-instalação	A	8	
andar-instalação	R	2	
sala-instalação	R	4	

Figura D.1 - ATRIBUTOS DA RELAÇÃO AQUISIÇÃO

NOME-ATRIBUTO	TIPO	COMPRIMENTO	COMENTÁRIO
num-aquisição	N	5	índice primário
sigla-tipo-equipamento	R	4	índice secundária
quantidade-configuração	N	2	

Figura **D.2 - ATRIBUTOS DA RELAÇÃO CONFIGURAÇÃO-AQUISIÇÃO**

NOME-ATRIBUTO	TIPO	COMPRIMENTO	COMENTÁRIO
num-aquisição	N	5	\ índice primária
cod-evento	N	2	/
sigla-orgão-despacho	A	4	índice secundário
data-entrada	N	6	
data-saida	N	6	
num-memo-despacho	N	6	
dest-memo-despacho	A	4	

Figura D.3 - ATRIBUTOS DA RELAÇÃO DESPACHO-AQUISIÇÃO

NOME-ATRIBUTO	TIPO	COMPRIMENTO	COMENTÁRIO
cod-evento	N	2	índice primário
situação	A	37	

Figura 0.4 - **ATRIBUTOS DA RELAÇÃO EVENTO**

NOME-ATRIBUTO	TIPO	COMPRIMENTO	COMENTÁRIO
sigla-fabricante	A	15	índice primário
nome-fabricante	A	45	

Figura D.5 - ATRIBUTOS DA RELAÇÃO FABRICANTE

NOME-ATRIBUTO	TIPO	COMPRIMENTO	COMENTÁRIO
sigla-fornecedor	A	15	índice primário
nome-fornecedor	A	45	

Figura D.6 - ATRIBUTOS DA RELAÇÃO FORNECEDQR

NOME-ATRIBUTO	TIPO	COMPRIMENTO	COMENTÁRIO
cod-orgão	N	3	índice primário
sigla-orgão	A	4	índice secundário
descrição-orgão	A	60	
orgão-superior	N	3	índice secundário
empresa	A	10	índice secundário

Figura D.7 - ATRIBUTOS DA RELAÇÃO ORGÃO

NOME-ATRIBUTO	TIPO	COMPRIMENTO	COMENTÁRIO
código-prédio	N	2	índiça primário
sigla-prédio	A	8	índice secundário
descrição-prédio	A	38	

Figura 5.8 - ATRIBUTOS DA RELAÇÃO PRÉDIO

NOME-ATRIBUTO	TIPO	COMPRIMENTO	COMENTÁRIO
sigla-tipo-equipamento	A	4	\
sigla-fornecedor	A	15	-> índice primário
sigla-fatricanta	A	15	/
data-última-compra	N	6	

Figura D.9 - DESCRIÇÃO DOS ATRIBUTOS DA RELAÇÃO PRODUTO

NOME-ATRIBUTO	TIPO	COMPRIMENTO	COMENTÁRIO
sigla-tipo-equipamento	A	4	índice primário
descrição-equipamento	A	50	

Figura D.10 - ATRIBUTOS DA RELAÇÃO TIPO-EQUIPAMENTO

APÊNDICE E

RESULTADOS DE DESEMPENHO DAS TRANSAÇÕES II NOS SGBD's

Conforme pode ser observado no capítulo V, os benchmarks de baixo nível foram executados várias vezes para gerar parâmetros dinâmicos corretos e seguros. Assim, executou-se n vezes Ti e analisou-se os resultados para cada SGBD. Os dados (em segundos) dos quadros abaixo permitiram calcular os outros parâmetros dinâmicos definidos na metodologia

A fim de facilitar a apresentação dos quadros, estes serão relatados seguindo a ordem de apresentação dos produtos no capítulo IV. A sequência será estritamente alfabética, não havendo nenhuma outra conotação na sua precedência.

Ti/N	Ti	5Ti	10Ti	100Ti	1000Ti
T1a	6,0	1,6	1,1	0,55	0,486
T5b	7,0	2,0	1,5	0,87	0,807
T2	7,0	2,0	1,5	0,94	0,878
T3	10,0	4,8	3,7	2,55	2,317
T4	8,0	2,2	1,5	0,81	0,745
T5	10,0	4,2	3,4	2,60	2,325
T6	12,0	5,0	4,1	3,16	2,985
T7	11,0	3,6	2,9	1,90	1,792
T8	9,0	3,2	2,5	1,67	1,593
T9	18,0	10,0	8,7	7,24	6,973
T10	5,0	1,8	1,3	0,87	0,801
T11	5,0	1,8	1,3	0,87	0,801
T12	9,0	3,0	2,2	1,47	1,371
T13	2,0	0,8	0,7	0,42	0,403
T14	19,0	10,4	8,9	7,33	6,981
T15	10,0	5,0	3,9	2,70	2,527
T16	20,0	11,4	9,8	8,21	7,912
T17	14,0	6,4	5,0	3,74	3,417
T18	19,0	8,4	6,7	4,94	4,678

Timprodutividade = 9s

Figura E,% - DESEMPENHO DAS TRANSAÇÕES TI NO COPPEREL

r'i/N	Ti	53i	10Ti	1ǫ0Ti	1000Ti
T1a	5,0	0,4	8▪3	0,93	0,143
T1b	3,0	0,8	0,7	0,67	0,603
T2	1,0	0,6	0,6	0,57	0,532
T3	6,0	6,6	10,7	15,23	28,920
T4	5,0	4,4	4,3	4,22	4,026
Y5	—	—	—	—	—
T6	—	—	—	—	—
T7	9,0	8,0	7,7	7,44	7,286
T8	3,0	2,2	ī,9	1,73	5,593
T9	81,0	80,0	78,4	76,96	75,844
T10	5,0	0,6	0,5	0,42	0,368
T11	3,0	5,4	1,2	ī,54	1,111
T12	2,0	0,8	0,7	8,64	8,6163
T13	3,0	1,4	ī,2	1,05	0,979
T14	82,0	79,6	77,9	76,84	75,830
T15	3,0	3,8	3,0	3,00	3,000
T16	134,0	133,8	133,7	133,64	133,615
T17	—	—	—	—	—
T18	4,0	3,6	3,4	3,27	3,251

Timprodutividade = 3s

Figura E.2 - DESEMPENHO DA8 TRANSAÇÕES T I NO DBASE

Ti/N	Ti	5Ti	10Ti	100Ti	1000Ti
T1a	1,0	0,4	0,3	0,24	0,235
T1b	2,0	0,6	0,6	0,50	0,468
T2	1,0	0,6	0,6	0,50	0,487
T3	4,0	4,2	6,7	10,87	15,171
T4	6,0	5,8	5,7	5,68	5,672
T5	—	—	—	—	—
T6	—	—	—	—	—
T7	5,0	4,0	3,9	3,89	3,881
T8	2,0	1,6	1,6	1,60	1,600
T9	54,0	53,4	53,2	53,07	53,018
T10	1,0	0,6	0,4	0,27	0,266
T11	1,0	1,0	0,9	0,85	0,811
T12	1,0	0,6	0,6	0,60	0,600
T13	1,0	1,0	0,8	0,65	0,641
T14	54,0	53,4	53,2	53,07	53,018
T15	2,0	1,4	1,4	1,40	1,400
T16	72,0	72,0	72,0	72,00	71,999
T17	—	—	—	—	—
T18	4,0	3,0	3,0	3,00	2,999

Timprodutividade = 5s

Figura E.3 - DESEMPENHO DAS TRANSAÇÕES TI NO DIALOG

i	i	Ti	5T*i*	10T*i*	100Ti	1000Ti
T1a		3,0	0,8	0,6	0,57	0,563
T1b		4,0	3,2	3,2	3,20	3,200
T2		3,0	2,4	2,4	2,40	2,400
T3		19,0	17,6	17,4	17,35	17,343
T4		3,0	2,2	2,1	2,01	1,995
T5		14,0	13,8	13,7	13,68	13,678
T6		1,0	0,6	0,5	0,41	0,380
T7		355,0	348,0	348,0	347,60	347,580
T8		11,0	8,0	7,9	7,74	7,727
T9		7,0	4,4	4,3	4,24	4,235
T10		2,0	0,8	0,6	0,50	0,451
T11		3,0	1,0	0,8	0,76	0,745
T12		2,0	1,0	1,0	0,92	0,917
T13		2,0	0,8	0,7	0,62	0,611
T14		3,0	1,8	1,6	1,47	1,454
T15		11,0	10,4	10,3	10,22	10,213
T16		3,0	1,8	1,6	1,58	0,577
T17		5,0	4,6	4,3	4,23	4,222
T18		16,0	15,0	14,9	14,76	14,743

Timprodutividade = 12s(carga ORACLE) + 6s(carga SQL)
= 18s

Figura E.4 - DESEMPENHO DAS TRANSAÇÕES T I NO QNACLE

Ti/N	Ti	5Ti	10Ti	100Ti	1000Ti
T1a	1,0	0,4	0,4	0,40	0,400
T1b	2,0	1,2	1,2	1,20	1,200
T2	—	—	—	—	—
T3	4,0	2,0	1,7	1,46	1,321
T4	—	—	—	—	—
T5	3,0	1,4	1,0	0,89	0,817
T6	—	—	—	—	—
T7	—	—	—	—	—
T8	—	—	—	—	—
T9	—	—	—	—	—
T10	—	—	—	—	—
T11	—	—	—	—	—
T12	—	—	—	—	—
T13	2,0	0,8	0,7	0,59	0,526
T14	—	—	—	—	—
T15	5,0	2,4	2,1	2,00	1,961
T16	—	—	—	—	—
T17	—	—	—	—	—
T18	—	—	—	—	—

Timprodutividade = 7s

Figura E.5 - DESEMPENHO DAS TRANSAÇÕES T I NO PARADQX

Ti/N	Ti	5Ti	10Ti	100Ti	1000Ti
Tia	1,0	0,6	0,4	0,20	0,180
T1b	2,0	1,8	1,6	1,41	1,357
T2	2,0	8,8	1,7	1,63	1,592
T3	5,0	4,8	4,6	4,57	4,563
T4	4,0	3,6	3,4	3,39	3,387
T5	3,0	2,6	2,4	2,28	2,137
T6	—	—	—	—	—
T7	35,0	34,0	33,8	33,65	33,527
18	3,0	2,6	2,4	2,22	2,138
T9	6,0	5,8	5,7	5,69	5,674
T10	2,0	1,6	1,4	1,17	1,158
T1	3,0	1,6	1,1	0,46	0,957
T12	2,0	0,8	0,7	0,68	0,563
T13	3,0	1,6	1,1	0,96	0,956
T14	6,0	5,8	5,7	5,69	5,676
T15	9,0	8,8	8,4	8,27	8,237
T16	8,0	7,6	7,4	7,29	7,273
7'17	7,0	6,0	5,9	5.87	5,853
T18	7,0	6,0	5,9	5,87	5,861

Timprodutividada = 6ir;

Figura E.6 - DESEMPENHO DAS TRANSAÇÕES TI NO RBASE

Ti/N	Ti	5Ti	10Ti	100Ti	1000Ti
T1a	1,0	0,8	0,7	0,60	0,613
T1b	1,0	1,0	1,8	0,96	0,957
T2	—	—	—	—	—
T3	2,0	1,4	1,4	1,38	1,378
T4	2,0	1,4	1,4	1,38	1,378
T5	2,0	1,4	1,4	1,38	1,378
T6	—	—	—	—	—
T7	—	—	—	—	—
T8	3,0	2,0	2,0	2,00	2,006
T9	3,0	3,2	3,2	3,20	3,201
T10	2,0	1,6	1,5	1,45	1,433
T15	2,0	1,0	1,0	1,88	1,001
T12	—	—	—	—	—
T13	2,0	1,4	1,3	1,29	1,285
T14	2,0	2,0	1,9	1,85	1,823
T15	2,0	1,6	1,5	1,45	1,433
T16	3,0	2,6	2,5	2,48	2,375
T17	—	—	—	—	—
T18	—	—	—	—	—

Timprodutividade = 7s

Figura E.7 - DESEMPENHO DAS TRANSAÇÕES TI NO ZIM

Conforme pode ser observado ria capítulo V, os benchmarks de alto nível T e Tmin foram executados várias vezes para confirmar os parâmetros dinâmicos. Assim, executou-se n veres V e Tmin, analisou-se os resultados para cada SGBD e observou-se o comportamento de T e Tmin em cada produto.

Os dados (em segundos) dos quadros abaixo permitiram calcular os outros parâmetros dinâmicos especificados na metodologia, conforme pode ser observado no capítulo V.

SGBD/N	Ti	5Ti	10Ti	100Ti	1000Ti
COPPEREL	213,0	173,7	142,9	115,94	113,102
DBASE	1213,0	1163,6	1166,9	1188,47	1223,735
DIALOG	883,0	834,4	835,3	847,38	861,773
ORACLE	485,0	437,3	399,9	395,87	393,983
PARADOX	71,0	66,8	62,2	58,63	57,549
RBASE	257,0	236,5	218,4	208,18	206,836
ZIM	108,0	91,5	86,2	82,97	81,878

Figura E.8 - DESEMPENHO DE T NOS SGBD's

SGBD/N	Ti	5Ti	10Ti	100Ti	1000Ti
COPPEREL	86,0	69,3	58,7	53,68	53,243
DBASE	319,0	311,8	314,9	318,38	326,576
DIALOG	214,0	198,9	201,7	206,23	216,239
QRACLE	352,0	336,7	328,8	323,57	321,531
PARADOX	65,0	61,3	56,4	53,85	53,427
RBASE	148,0	130,1	118,7	113,19	112,999
ZIM	89,0	67,3	56,9	53,68	53,666

Figura E.9 - DESEMPENHO DE Tmin NOS SGBD's