Universidade Federal da Paraíba
Centro de Ciências e Tecnologia
Coordenação de Pós-Graduação em Informática

Proposta e Implementação de um Modelo Cliente-Servidor para Administração de Bancos de Dados em Sistemas Distribuídos

Edjander de Souza Mota

Campina Grande

Junho - 1994

Edjander de Souza Mota

Proposta e Implementação de um Modelo Cliente-Servidor para Administração de Bancos de Dados em Sistemas Distribuídos

Dissertação apresentada ao Curso de Mestrado EM INFORMÁTICA da Universidade Federal da Paraíba, em cumprimento às exigências para obtenção do Grau de Mestre

Área de Concentração: Ciência da Computacão

Maria de Fátima Q. Vieira Turnell
(Orientadora)

Campina Grande

M917p Mota, Edjander de Souza.
Proposta e implementação de um modelo cliente-servidor para administração de bancos de dados em sistemas distribuídos / Edjander de Souza Mota. - Campina Grande, 1994.
71 f.

Dissertação (Mestrado em Informática) - Universidade Federal da Paraíba, Centro de Ciências e Tecnologia, 1994.
"Orientação : Profa. Dra. Maria de Fátima Queiroz Vieira Turnell".
Referências.

1. Banco de Dados - Sistema Distribuído - Modelo Cliente-Servidor. 2. Dissertação - Informática. I. Turnell, Maria de Fátima Queiroz Vieira. II. Universidade Federal da Paraíba - Campina Grande (PB).

CDU 004.65(043)

Proposta e Implementação de um Modelo Cliente-Servidor para Administração de Bancos de Dados em Sistemas Distribuídos

Edjander de Souza Mota

DISSERTAÇÃO APROVADA EM 24 / 06 / 94

Maria de Fátima Q. Vieira Turnell

MARIA DE FÁTIMA Q. VIEIRA TURNELL (PhD)
(ORIENTADORA)

Ana Carolina Salgado

ANA CAROLINA SALGADO (Dr.)
(COMPONENTE DA BANCA)

JOBERTO S. B. MARTINS (Dr.)
(COMPONENTE DA BANCA)

GEDEON J. DOS SANTOS FILHO (MSc.)
(COMPONENTE DA BANCA)

CAMPINA GRANDE - PB
JUNHO - 1994

DEDICATÓRIA

Aos meus pais pelo esforço dedicado, renúncia e acima de tudo pela minha formação moral, intectual e cristã.

Aos meus irmãos Edjair, Edjard, Sheila e Amélia, pelo incentivo e uma vida repleta de harmonia e amor.

Aos meus filhos Lívia e Leandro, uma alegria constante em nosso lar.

A você Arlene, pelo companheirismo, amor e dedicação, nos momentos díficeis aqui em Campina Grande.

AGRADECIMENTOS

A Deus.

A minha orientadora Fátima Turnell, pelas demonstrações de confiança, amizade e competência.

Ao meu amigo de infância Gedeon Filho, pela orientação em algumas fases deste trabalho. Espero que seu ditado venha a se confirmar após a defesa desta tese.

A Claudio Monteiro (Pará), pelas contribuições no decorrer deste trabalho, e a demonstração de amizade sincera.

Ao amigo Fábio Nobrega, que muitas vezes compartilhou comigo o único PC do departamento de Minas, e pelas conversas e piadas memoráveis.

A Edwin Aldrin e Gledson Maia, amigos para sempre.

Aos amigos Kátia, Kíssia, Eliane, Carlos, Joseana, Iana e Marckson. Sempre lembrarei de vocês.

Ao Paraibano Hércules Gomes Pimentel, que revolucionou o departamento de informática da FUCAPI.

Aos pessoal do NPD do campus II, Jonas, Elisa, Miguel, Costa e Dênis, pela ajuda inestimável.

Aos amigos de luta, desde os tempos de faculdade, Sidney Paulo e Ernande Melo.

A minha querida professora Maria, hoje telefonista da FUCAPI. A casa da amizade jamais será esquecida, que saudades daqueles tempos!

Aos amigos Moreira e Robson Pequeno do DEE, pelos momentos de descontração e sincera amizade.

Ao grande Alberto José F. de Lima, pela ajuda inestimável na editoração deste trabalho.

A Ana Lúcia Guimarães, por seu trabalho competente, presteza e amizade.

Aos demais professores e bibliotecárias.

A todos os colegas do curso de mestrado, pela solidariedade nos momentos difíceis.

A FUCAPI, por ter me proporcionado esta oportunidade de aprimoramento.

Ao CNPq pelo apoio financeiro.

Muito Obrigado

Em memória
a Antonio Zenir Faraco Picanço Junior, onde estiver serás sempre lembrado por mim e nossa família. Você é massa cara.

SUMÁRIO

LISTA DE FIGURAS

RESUMO

As ferramentas do mercado atual voltadas para o trabalho de administração de banco de dados, não atendem as necessidades de gerência. Uma nova classe de sistemas voltados para esse tipo de tarefa, está surgindo para suprir tais necessidades. Estes sistemas utilizam os conceitos de banco de dados Cliente - Servidor, que se aplicam em sistemas distribuídos heterogêneos.

Este trabalho apresenta o projeto e implementação de uma ferramenta baseada no modelo conceitual Cliente - Servidor, que auxiliará DBAs (Administradores de Banco de Dados) nas suas atividades de gerência. A ferramenta que tem como característica, a administração de banco de dados situados localmente ou remotamente, em uma rede heterogênea.

CAPÍTULO I

INTRODUÇÃO

As principais atividades de um DBA (Administrador de Banco de Dados), estão relacionadas com a segurança, a manutenção dos objetos que compõem o banco de dados (tabelas, visões, índices e aplicações) e a monitoração de desempenho.

Dentre essas atividades descritas anteriormente, o DBA convive quase que diariamente com tomadas de decisão quanto à parâmetros que afetam a eficiência do banco de dados (BD). São decisões quanto à utilização de espaço físico, aos métodos de acesso aos dados, à frequência e tipo de *backups*; passando pela segurança, em vários níveis, no controle de acesso tanto pelo usuário quanto pela aplicação.

Destaque-se ainda a frequente demanda imposta pelas solicitações dos usuários do banco (analistas e programadores), para verificar as possíveis causas de degradação de uma aplicação, ou até mesmo para criação de novas tabelas e índices.

Embora conte com um número bastante elevado de aplicativos para gerência, isolados ou integrados no SGBD (Sistema gerenciador de banco de dados), que o auxiliem neste processo, o volume de decisões a serem tomadas pelo DBA exige uma agilidade na busca de causas e possíveis soluções para os problemas emergentes que transcende à capacidade dos recursos existentes.

Um exemplo da complexidade desta tarefa, são os vários fatores que podem levar à queda de desempenho de uma aplicação. Dentre as possíveis causas temos o tamanho de *buffers*, os *deadlocks* [1] , área física insuficiente, *locks* em caso de atualização, entre outras. É necessário portanto analisar os dados que são lidos pela aplicação. Cita-se como exemplo, os BDs relacionais onde as tabelas. podem estar com seus índices fragmentados ou alguma delas até mesmo sem índice definido. Sendo portanto necessário reorganizar os índices destas tabelas, ou criar um índice que seja capaz de melhorar o desempenho da aplicação.

Para que a eficiência da tarefa de administração de banco de dados possa aumentar, é necessário uma ferramenta que permita ao DBA recuperar as informações para análise de forma simples e dentro de uma representação lógica que agrupe os dados relevantes para uma decisão em uma mesma tela. A representação gráfica de alguns índices de desempenho também aceleraria a tomada de decisões. E finalmente, um recurso que permitisse consultar mais de um banco de dados instalados em diferentes máquinas de uma rede, em uma mesma tela e ambiente de software, permitiria ao DBA atender às demandas de uma de suas tarefas mais frequentes que consiste em atender às solicitações dos usuários destes bancos para averiguar as causas de degradação de desempenho e aplicar as devidas correções.

Baseado neste contexto, e na tendência atual em distribuir recursos, devido ao crescimento de sistemas de rede, foi concebido um sistema que pudesse auxiliar administradores de banco de dados em suas tarefas de gerência - o DBA-CS. Este sistema utiliza uma interface gráfica amigável, que facilita o acesso aos recursos de um SGBD relacional, e a avaliação de desempenho dos diversos aplicativos que manipulam o banco de dados.

O DBA-CS permite que o DBA gerencíe bancos de dados remotamente podendo estar distribuídos em máquinas com arquiteturas de hardware diferentes. Para que a comunicação entre essas máquinas fosse possível, foi utilizada a pilha de protocolos TCP/IP, como base de comunicação entre máquinas heterogêneas [Commer 93]. Esta arquitetura de comunicação é inerente às plataformas de hardware e, tornou um padrão ***de facto*** no cenário internacional devido suas características acomodarem as várias tecnologias de rede [Derfler 93].

---

[1] Deadlock significa impasse, ou seja, quando existe um conjunto de transações tal que cada conjunto no

A Figura 1.1 mostra uma possível distribuição de BDs em um conjunto de redes.

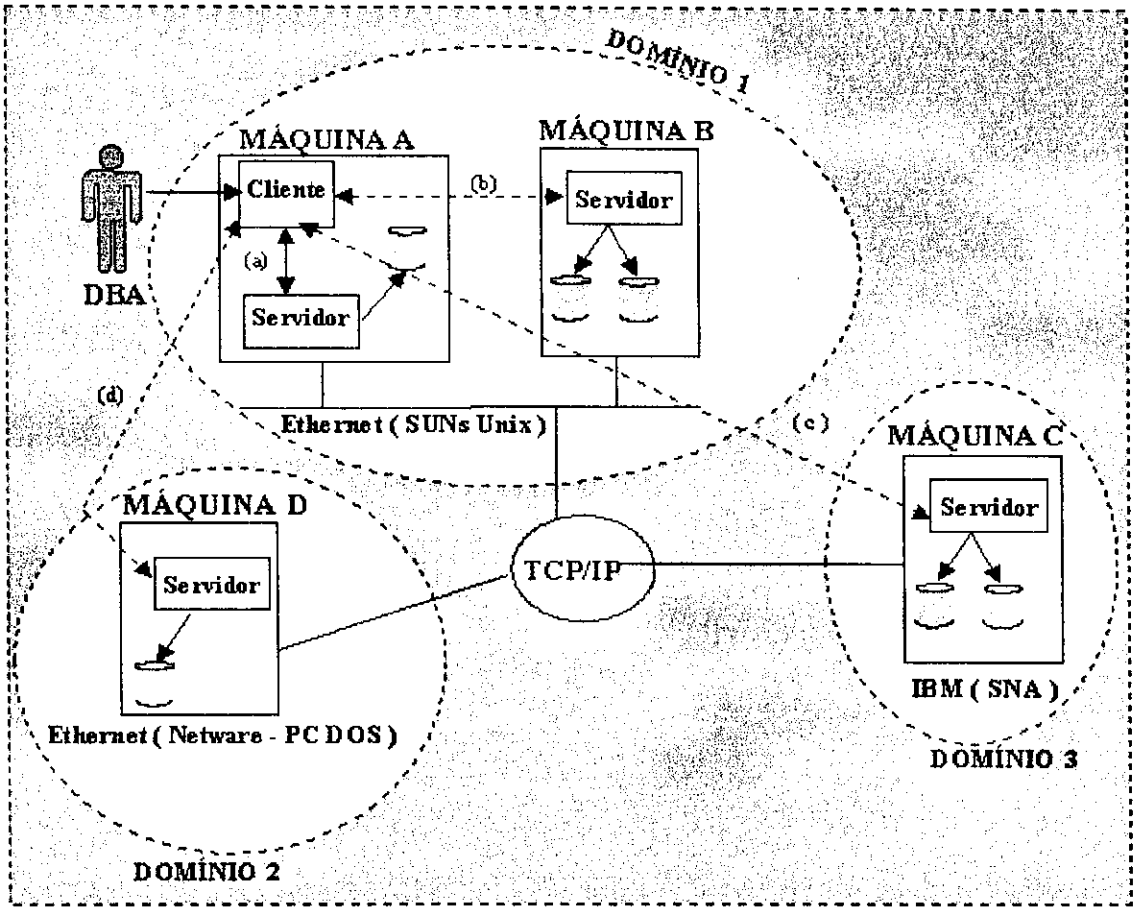

Figura 1. Distribuição de BDs, em três domínios[2] .

O sistema DBA-CS, é composto por um conjunto de processos **Servidores** e um processo **Cliente,** a serem discutidos na seção 2.2.2.

A Figura 1.1 ilustra que o DBA administrará BDs locais, no domínio 1, através da relação Cliente-Servidor como mostra a Fig. 1.1- a , e a Figura 1.1- b.

A partir do domínio 1 também poderá administrar os BDs nos domínios 2 e 3, através das relações: Fig. 1.1- d e Fig. 1.1- c, respectivamente.

[2] Neste trabalho, domínio é entendido como sendo o poder de administração do DBA em relação aos BDs

Para validar os testes da ferramenta, utilizamos o SGBD relacional SQL/DS (Structured Query Language/Data System) da IBM, que utiliza a linguagem SQL estendida, para definição e manipulação de dados[IBM 91b]. Apesar de usarmos um gerenciador específico, o sistema pode facilmente se adequar a outros SGBDs relacionais.

1.1 Objetivos

1.1.1 Objetivo geral

Este trabalho objetiva apresentar o projeto e implementação do DBA-CS, um modelo Cliente-Servidor para administração de bancos de dados em sistemas distribuídos, que se propõe auxiliar os DBAs na tarefa de gerência de um ou mais banco de dados.

1.1.2 Específicos

Os objetivos específicos deste trabalho são:

- Estruturar as principais atividades de um DBA, em um ambiente único, procurando facilitar seu trabalho de gerência.

- Permitir que bancos de dados remotos sejam gerenciados em uma mesma tela, independente de sua localização ou plataforma de hardware.

1.2 Trabalhos relacionados

Existem alguns produtos no mercado de banco de dados Cliente-Servidor (C-S), dentre eles a Oracle Server 6.0, que permite a comunicação C-S utilizando o TCP/IP, para todas as versões do produto (em PC, UNIX e *Mainframes*). É possível o acesso a banco de dados (BD) não Oracle, utilizando como por exemplo, o utilitário SQL*Connect, para SGBDs SQL/DS e DB2 da IBM.

O acesso a BDs remotos, está limitado apenas a consultas, mais está prometida para 94, a versão 7.0, que atualizará o software Oracle com os mais recentes avanços na tecnologia de BDs C-S. O *front-end* do produto ainda utiliza uma interface de linha de comando para administrar e consultar o BD.

Outro produto similar ao Oracle, é o INGRES, que é executado em plataformas UNIX ou VAX/VMS. O utilitário INGRES/Star que é acoplado ao INGRES, tem suporte a processamento distribuído, em versões intermediárias do produto e com outros BDs VAX/VMS ou baseados em *mainframes*. A ferramenta de administração de banco de dados tem uma interface baseada em caractere, utilizando a linguagem SQL para manipulação de BDs, também oferece suporte para as linguagens QUEL e EQUEL.

O DBA-CS pode ser comparado aos utilitários SQL*Connect da Oracle e ao INGRES/STAR do INGRES, contudo, o DBA-CS difere desses produtos, por se tratar de uma ferramenta voltada exclusivamente para a administração de banco de dados.

Podemos destacar como uma das principais vantagens do DBA-CS, a sua interface gráfica, que permite uma interação mais amigável entre o DBA e a ferramenta proposta, além de aumentar a sua produtividade. Outro fator que diferencia O DBA-CS das outras ferramentas citadas acima, é a possibilidade de este sistema gerenciar vários bancos de dados, que estejam distribuídos em máquinas com arquiteturas de hardware diferentes.

1.3 Organização do trabalho

Neste capítulo foi apresentada uma visão geral da ferramenta proposta, assim como seus objetivos e trabalhos relacionados.

No capítulo 2, é apresentado o embasamento teórico que deu suporte ao desenvolvimento deste trabalho. Uma visão sobre as principais atividades de um DBA no trabalho de gerência e, a infra-estrutura de comunicação para desenvolvimento de aplicações em ambientes distribuídos.

No capítulo 3, é apresentada a arquitetura do DBA-CS e, os módulos que a compõem: Cliente (Interface gráfica e módulo de rede), Servidor (módulo de rede e os principais serviços disponíveis).

No capítulo 4, são feitas as considerações de implementação do DBA-CS, e apresentadas a infra-estrutura de desenvolvimento e os requisitos funcionais de rede.

O capítulo 5, apresenta as conclusões e sugestões para trabalhos futuros.

O Apêndice A, é o manual do usuário, que serve de referência para usuários do sistema.

CAPÍTULO II

REFERENCIAL TEÓRICO

Neste capítulo, é feita uma abordagem dos assuntos que formarão a base para o desenvolvimento do DBA-CS. Inicialmente é dada uma visão sobre a atividade de administração de banco de dados, destacando-se a pessoa do DBA e suas responsabilidades, seguida da descrição da infra-estrutura de comunicação utilizada neste projeto.

2.1 Administração de Banco de Dados

A atividade de administrar, surgiu da necessidade das organizações manterem adequadamente seus recursos, de forma a alcançarem seus objetivos. O processo administrativo envolve planejamento, organização, liderança e controle do desempenho das pessoas organizadas para uma determinada tarefa [Hampton 83].

Alguns estudiosos das teorias administrativas, consideravam pessoas, capital e materiais, como os únicos recursos (bens) de uma organização, no entanto, um recurso tão antigo quanto os demais, tornou-se precioso e imprescindível, no mercado competitivo em que vivemos; a **informação**.

O surgimento da tecnologia de banco de dados, contribuiu para um melhor uso da informação, através do controle centralizado dos dados [Date 91].

Para que o controle dos dados operacionais de uma organização, fossem mais eficientes, foi criada uma nova atividade nos centros de processamento de dados; a administração de banco de dados. Esta atividade é controlada por uma pessoa (ou

um grupo), denominado Administrador de Banco de Dados (**DBA**). Este grupo é responsável pela gerência, manutenção e suporte aos usuários do BD.

Nas próximas secções serão apresentados o DBA e suas responsabilidades, entre: Segurança, Manutenção e Desempenho.

2.1.1 O DBA e suas Responsabilidades

O DBA como foi descrito anteriormente, é a pessoa ou grupo de pessoas que exerce efetivamente a administração do banco de dados, ele é o responsável pelos dados, não o seu dono. O DBA trabalha em função dos usuários do BD, programadores, analistas, gerentes e equipe de operação. O DBA precisa manter uma relação estreita e efetiva com todos os usuários do banco de dados, para captar suas necessidades, e oferecer subsídios para que estes, executem eficientemente suas tarefas[Atre 80].

As principais responsabilidades de um DBA, estão relacionadas com a Segurança, a Manutenção dos objetos que compõem o BD (tabelas, visões, índices e aplicações) e a monitoração de sua performance. Estes três aspectos serão discutidos a seguir.

2.1.2 Segurança

Uma das principais responsabilidades de um DBA, refere-se ao controle de acesso aos dados protegendo-os contra acessos indevidos e à definição de estratégias para a recuperação do BD, em caso de falha de hardware ou erro humano a partir da codificação de rotinas de backup [Korth 89]. Os dados operacionais de uma organização, como foi descrito na seção 2.1, constituem-se em um bem dos mais valiosos, sendo assim, a segurança e a integridade desses dados são essenciais para quem os manipula.

Prover segurança é garantir que dados sigilosos sejam manipulados apenas por pessoas autorizadas, e dentro de um controle pré-definido. A manutenção dos objetos do BD, é tarefa constante em todo o ciclo de vida do BD, como consequência a monitoração de desempenho também deve ser permanente, com a finalidade de manter o bom funcionamento do BD.

Existem vários níveis de segurança que devem ser considerados, para a obtenção de um grau aceitável de proteção aos dados uma vez que é impossível conseguir segurança total.

Em geral, a equipe que gerencía o BD, trabalha em conjunto com a equipe de desenvolvimento e os vários departamentos da organização, a fim de decidirem as estratégias de segurança, ou seja, definir quais as pessoas autorizadas a manipular os dados do BD.

Os usuários podem manipular esses dados de duas formas: através de programas de aplicação, desenvolvidos pela equipe de desenvolvimento ou através de linguagens interativas, que acompanham os SGBDs.

- As linguagens interativas oferecem um conjunto de instruções, que podem ser manipuladas de acordo com as necessidades dos usuários. A Linguagem SQL (*Structured Query Language*) disponível para a maioria de BDs relacionais é um exemplo [Date 91]. Essas linguagens permitem ao DBA, diferenciar categorias de usuários, ou seja, usuários com manipulação de leitura, outros de leitura e gravação e assim por diante. A definição dessas categorias, devem ser criteriosas, pois instruções mal utilizadas afetam diretamente os dados do BD .

- Por outro lado, os programas de aplicação são codificados, utilizando uma linguagem de programação (Cobol, Fortran, C). As instruções de manipulação da linguagem SQL por exemplo, podem ser embutidas nesses programas para que se possa acessar o BD.

O DBA deve especificar a abrangência de acesso aos dados, tanto para os programas de aplicação, quanto para os usuários que utilizam uma linguagem

interativa. Os acessos podem ser atribuídos em níveis ou tipo, como segue: autorização de leitura, inserção, atualização e eliminação de dados.

Os níveis podem ser combinados, para permitir um maior grau de acesso, de acordo com as necessidades de cada usuário ou programa de aplicação.

A linguagem SQL, possui mecanismos para que se possa implementar as estratégias de segurança. Um desses mecanismos consiste das visões, que restringem os acessos de usuários, a pequenas porções de uma ou mais tabelas [Date 91]. Esses mecanismos no entanto, não definem que operações podem ser aplicadas sobre visões ou tabelas. As instruções *grant* e *revoke* complementam essas especificações. A primeira define os tipos de acessos permitidos, a segunda os remove.

Outra medida de segurança que deve ser observada pelo DBA , consiste em codificar rotinas que realizem *backups d*e tabelas. Estas rotinas indispensáveis são para a restauração parcial ou total desses objetos, com uma periodicidade definida, de acordo com a intensidade de atualização sobre as tabelas, ou a pedido do usuário.

2.1.3 Manutenção

Na prática são discutidas duas fases que antecedem a manutenção do BD, que são: a instalação do SGBD e o projeto lógico do banco de dados.

O termo manutenção, refere-se às alterações que um banco de dados sofre, desde sua criação [Atre 80]. A primeira tarefa de um DBA, é especificar os elementos tais como: volume dos discos e espaço de armazenamento, separação física e exigências de afinidades. Esse trabalho de nível interno, é essencial para o bom desempenho do SGBD [Date 91].

A segunda fase está relacionada com o processo de modelagem do BD. O DBA deve ter uma visão geral da organização, para identificar o relacionamento e o fluxo de informação entre os vários departamentos [Atre 80]. Em seguida, com as prioridades da organização, deverá especificar o projeto lógico, onde serão definidas as entidades e seus relacionamentos, que poderão tornar-se tabelas.

E finalmente é feito o projeto físico, onde é especificado como os dados serão representados internamente. Os principais serviços que podem ser executados nesta fase são: criação, alteração e remoção de tabelas, índices e *views*; carga de dados em tabelas do BD. Essas operações geralmente são executadas, através de uma linguagem de definição de dados (DDL). A linguagem SQL, oferece um conjunto de instruções que permite, o mapeamento do esquema conceitual para o esquema interno. As instruções de mapeamento dessa linguagem (DDL) são: *create table, create index, create view, drop table e drop index* [Date 91].

As fases um e dois, mostram que o DBA deve conhecer com uma certa profundidade, o equipamento de hardware e o sistema operacional onde será instalado o SGBG. E, principalmente, o funcionamento do gerenciador, desde a sintaxe dos comandos que manipulam o BD, até sua arquitetura, de modo a minimizar os problemas de desempenho, que serão discutidos da próxima secção.

2.1.4 Desempenho

O desempenho de um BD, está intrinsicamente ligado às definições dos objetos que o compõem, e de que forma eles serão manipulados.

Alguns fatores externos ao BD, podem causar um mau funcionamento do gerenciador, tais como: sobrecarga do sistema, problemas de comunicação, capacidade de armazenamento insuficiente etc [IBM 88].

Os dados de uma tabela podem ser manipulados por um programa de aplicação, ou por uma linguagem interativa, como foi descrito na seção 2.1.2.

As operações que mais sobrecarregam um SGBD, são aquelas que precisam fazer uma leitura sequencial em tabelas com muitas linhas, pois requerem constantes acessos a disco (E/S).

O DBA deve, portanto, interagir constantemente com a equipe de desenvolvimento, para conhecer suas necessidades de dados sobre o BD, e fornecer soluções adequadas a execução dos programas aplicativos [Atre 80].

Uma solução para o problema acima, seria definir uma área de página de *buffer*, com grandes proporções.

As operações que necessitarem manipular apenas uma linha de uma tabela, podem ser ineficientes, caso um índice não seja especificado para acelerar a recuperação da mesma. O DBA então deve especificar pelo menos um índice para a tabela [Date 91]. Esse índices podem tornar-se fragmentados, através de frequentes operações de inserção e remoção de linhas da tabela, implicando em queda de desempenho, portanto, o DBA deve monitorar constantemente o BD, e reorganizar as tabelas com índices fragmentados.

A monitoração de desempenho, certamente é um desafio aos DBAs, pois várias razões podem levar à queda de desempenho. As ferramentas de monitoração que acompanham os gerenciadores, muitas vezes são insuficientes ou não oferecem clareza nos seus resultados.

Um exemplo de um recurso bastante útil são os catálogos de um SGBD. Eles são uma espécie de dicionário de dados, armazenando informações sobre tabelas, aplicações e índices, que permitem ao DBA, acompanhar o desempenho do BD.

2.2 Infra-estrutura de Comunicação

O DBA-CS foi idealizado de forma a permitir a gerência de BDs em qualquer parte da rede. Esta rede pode ser local, a longa distância, heterogênea ou formada pela integração de várias outras redes.

Nas próximas seções serão discutidos, os aspectos conceituais para a implementação deste projeto. Primeiramente é apresentado o conjunto de protocolos TCP/IP, utilizado neste projeto como base de comunicação entre máquinas possivelmente heterogêneas; o modelo conceitual Cliente-Servidor e finalmente o modelo de chamada de procedimento remoto (RPC).

2.2.1 TCP/IP uma Arquitetura de Comunicação

O TCP/IP (*Transmission Control Protocol/Internet Protocol*) é um conjunto de protocolos, que permite que dados sejam transportados entre computadores com arquiteturas diferentes [Commer 91]. Sua grande aceitação no mercado, o tornou um padrão ***de facto.***

A arquitetura TCP/IP é composta por uma arquitetura em camadas, que pode ser comparada com três camadas do modelo OSI (*Open System Interconnection*): camada física, de rede e de aplicação, como mostra a Figura 2.1 [Tanenbaum 89]. O TCP/IP não especifica a camada física, portanto o usuário pode usar qualquer

camada física de transmissão, tanto para **LAN** (*Local Area Network*) como para **WAN** (*Wide Area Network*) [IBM 92].

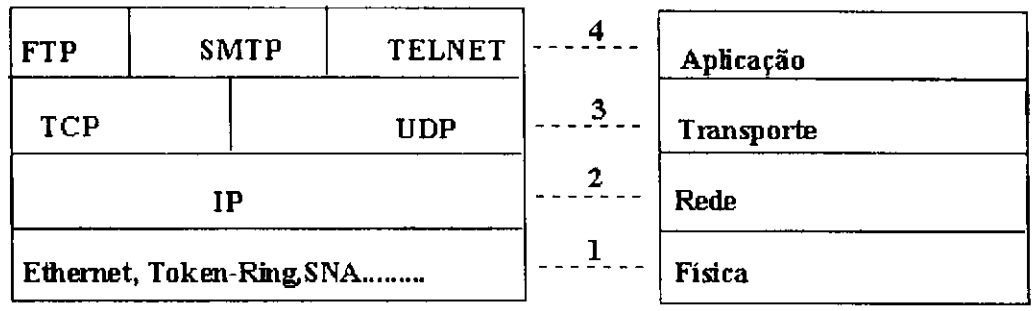

Figura 2.1 - Arquitetura TCP/IP x OSI

A camada de rede, IP (*Internet Protocol*), interconecta uma ou mais redes, ou seja, esta camada providencia o roteamento de mensagens de um dispositivo para outro, em um ambiente local ou sob uma rede a longa distância. Do ponto de vista do usuário, a parte visível do IP são os endereços que cada máquina participante da rede recebe. Tais endereços são únicos, e são chamados **endereços IP** [Moura 92].

O IP define um endereço *string* de 32 bits, para cada *HOST* da rede. Este endereço é constituído de dois campos: o identificador de rede (**NetID**) e o HOST equivalente da rede (**HOST ID**). O NetID é o mesmo para todos os HOSTS de uma determinada rede.

Por exemplo, o endereço 150.165.2.1 é assim descrito: O NetID é representado pelo número 150, enquanto que 165.2.1 identifica o HOST ID.

Apesar desses endereços fornecerem uma representação compacta e conveniente para especificar as máquinas as quais desejamos contactar, usuários preferem usar nomes (mnemônicos). No exemplo anterior o endereço 150.165.2.1 poderia ser codificado como magali.dsc.ufpb.br, que simplificaria o reconhecimento da rede e da máquina, por parte do usuário. A escolha de nomes deve ser única, para evitar ambiguidades.

A terceira camada oferece dois protocolos que transportam informações, entre nós da rede. O TCP (*Transmission Control Protocol*), orientado para conexão,

garante a transmissão ordenada e sem erro entre pares de processos em uma rede, é um protocolo confiável. O UDP (*User Datagrama Protocol*), é um protocolo sem conexão e sem confiabilidade, ou seja, garantia de entrega, mas obtém um bom desempenho, por não sofrer atraso inicial para o estabelecimento de conexão.

Na quarta camada estão os protocolos de alto nível, chamados de protocolo de aplicação, tendo como função básica, efetuar a interface com as aplicações que desejam utilizar os meios de comunicação de dados. Os protocolos primários do TCP/IP são TELNET, FTP e SMTP [IBM 92]. Outros protocolos poderão estar disponíveis em outras implementações TCP/IP [IBM 92].

2.2.2 Modelo Cliente-Servidor

Nesta seção será apresentado o modelo conceitual Cliente-Servidor, que foi utilizado para modelar o DBA-CS, e por conseguinte os tipos de servidores.

Como foi descrito na seção 2.2.1, uma das características do TCP/IP, é permitir que dois programas de aplicação estabeleçam uma conexão entre si, e transportem dados de um para o outro.

Apesar das características descritas anteriormente, o TCP/IP não especifica as regras de interação entre essa aplicações. O modelo Cliente-Servidor (C-S) serve de base para esse tipo de comunicação, especificando que um das aplicações espere indefinidamente, o contato da outra.

O modelo C-S especifica que dois processos distintos sejam cooperativos. O servidor é um processo que oferece serviços aos processos clientes [Commer 93].O servidor recebe um pedido de serviço, executa e retorna o resultado ao cliente. A Figura 2.2 mostra como funciona este esquema.

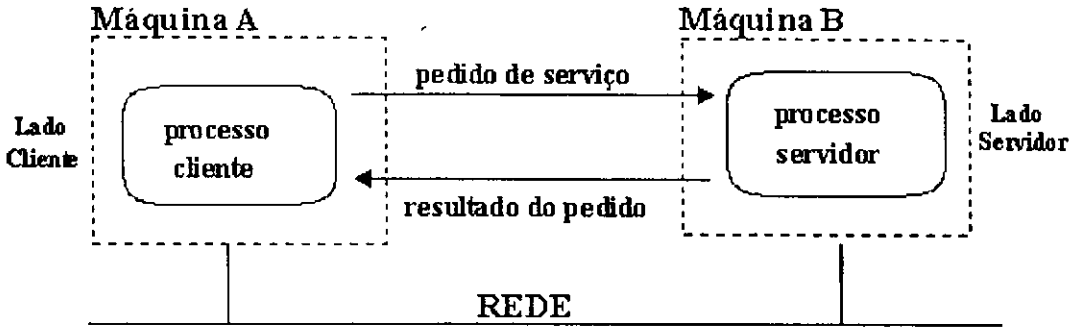

Figura 2.2 - Modelo Cliente-Servidor

Um dos aspectos importantes em uma comunicação C-S, esta relacionado com o tratamento oferecido pelo servidor, para pedidos de serviços por diversos usuários ao mesmo tempo. Eles são classificados quanto à estratégia de serviço; Servidor Iterativo e Servidor Concorrente; e quanto à manutenção de estado: Servidor com estado ou servidor sem estado [Filho 93].

- **Servidores Iterativos e Concorrentes**

Os servidores iterativos executam um único pedido por vez, ou seja, os pedidos são enfileirados e executados na ordem em que chegaram. Esse tipo de servidor é simples de ser implementado, em contra partida, pode causar uma demora considerável para enviar a resposta, dependendo do tipo de operação que se deseje fazer.

Já os servidores concorrentes, possuem a facilidade de atender a vários serviços simultaneamente, criando múltiplos *threads* [1] , um para cada pedido de serviço [Tanenbaum 89]. Esse tipo de servidor executa pedidos simultaneamente, quando a máquina possui múltiplos processadores, caso contrário a concorrência será apenas virtual.

[1] *Thread* representa o agente de controle de uma seqüência de instrucões sendo executado por um programa.

- **Servidores com e sem estado**

O servidor com estado tem como característica principal, manter alguma informação referente aos pedidos dos clientes para atendê-los corretamente [Filho 93]. Essas informações ou estados, são utilizadas pelo servidor, para dar continuidade a novos pedidos, sem que seja necessário enviar um conjunto de informações já armazenadas [Commer 93].

O servidor sem estado não mantém qualquer informação sobre os pedidos de clientes, o estado é mantido pelo próprio cliente, e sua maior vantagem é a tolerância a falhas[Commer 93].

2.2.3 Modelo de Chamada de Procedimento Remoto

Chamada de procedimento remoto, ou RPC (*Remote Procedure Call*), consiste numa biblioteca de funções que permite a comunicação entre processos, desenvolvidos em ambiente distribuído [Corbin 91]. O RPC estende os conceitos de uma chamada local para uma chamada remota, ou seja, funções locais podem ser executadas em outra máquina [Comer 93].

O funcionamento do RPC, está relacionado com o modelo Cliente-Servidor nos seguintes aspectos: procedimentos que serão executados em outras máquinas, são chamados de servidores, e aqueles que fazem a chamada destes procedimentos, são os clientes.

Para que processos remotos possam se comunicar, é necessário que o servidor exista e fique a espera de solicitações. Quando o cliente solicita um serviço, ele envia uma mensagem ao servidor, contendo os parâmetros do procedimento. Neste momento o controle de execução é suspenso e passado ao servidor. Este por sua vez, executa o serviço e retorna uma mensagem contendo a resposta. Ao receber a mensagem, o cliente passa a ser executado após a última instrução RPC. A Figura 2.3 ilustra este procedimento.

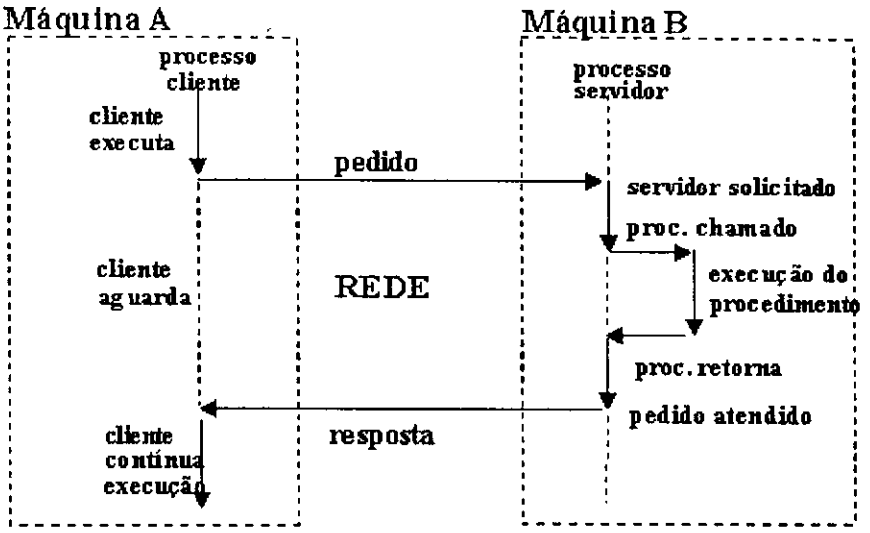

Figura 2.3 - Modelo de Chamada de Procedimento Remoto (RPC)

Algumas particularidades do RPC, tais como: passagem de parâmetros, variáveis globais, protocolo de transporte, entre outras, [Filho 93], devem ser conhecidas pelo programador, para que se obtenha um bom nível de programação.

A administração de banco de dados em geral está restrita a um tipo de SGBD específico, na seção 2.1, procuramos mostrar em linhas gerais, que independente do software que gerenciará o BD, as principais tarefas do dia a dia de um DBA, se resumem em segurança, manutenção e monitoração de desempenho. Foram encontradas algumas dificuldades para estruturar o texto, pois são poucas as referências que abordam especificamente a administração de BDs.

Este capítulo procurou fornecer um embasamento teórico, para situar o leitor nos limites da ferramenta proposta; as principais atividades de um DBA, e uma solução para comunicação de processos distribuídos em máquinas diferentes.

No próximo capítulo, será apresentado a arquitetura do DBA-CS, assim como, os detalhes de sua funcionalidade.

CAPÍTULO II

REFERENCIAL TEÓRICO

Neste capítulo, é feita uma abordagem dos assuntos que formarão a base para o desenvolvimento do DBA-CS. Inicialmente é dada uma visão sobre a atividade de administração de banco de dados, destacando-se a pessoa do DBA e suas responsabilidades, seguida da descrição da infra-estrutura de comunicação utilizada neste projeto.

2.1 Administração de Banco de Dados

A atividade de administrar, surgiu da necessidade das organizações manterem adequadamente seus recursos, de forma a alcançarem seus objetivos. O processo administrativo envolve planejamento, organização, liderança e controle do desempenho das pessoas organizadas para uma determinada tarefa [Hampton 83].

Alguns estudiosos das teorias administrativas, consideravam pessoas, capital e materiais, como os únicos recursos (bens) de uma organização, no entanto, um recurso tão antigo quanto os demais, tornou-se precioso e imprescindível, no mercado competitivo em que vivemos; a **informação**.

O surgimento da tecnologia de banco de dados, contribuiu para um melhor uso da informação, através do controle centralizado dos dados [Date 91].

Para que o controle dos dados operacionais de uma organização, fossem mais eficientes, foi criada uma nova atividade nos centros de processamento de dados; a administração de banco de dados. Esta atividade é controlada por uma pessoa (ou

um grupo), denominado Administrador de Banco de Dados (**DBA**). Este grupo é responsável pela gerência, manutenção e suporte aos usuários do BD.

Nas próximas secções serão apresentados o DBA e suas responsabilidades, entre: Segurança, Manutenção e Desempenho.

2.1.1 O DBA e suas Responsabilidades

O DBA como foi descrito anteriormente, é a pessoa ou grupo de pessoas que exerce efetivamente a administração do banco de dados, ele é o responsável pelos dados, não o seu dono. O DBA trabalha em função dos usuários do BD, programadores, analistas, gerentes e equipe de operação. O DBA precisa manter uma relação estreita e efetiva com todos os usuários do banco de dados, para captar suas necessidades, e oferecer subsídios para que estes, executem eficientemente suas tarefas[Atre 80].

As principais responsabilidades de um DBA, estão relacionadas com a Segurança, a Manutenção dos objetos que compõem o BD (tabelas, visões, índices e aplicações) e a monitoração de sua performance. Estes três aspectos serão discutidos a seguir.

2.1.2 Segurança

Uma das principais responsabilidades de um DBA, refere-se ao controle de acesso aos dados protegendo-os contra acessos indevidos e à definição de estratégias para a recuperação do BD, em caso de falha de hardware ou erro humano a partir da codificação de rotinas de backup [Korth 89]. Os dados operacionais de uma organização, como foi descrito na seção 2.1, constituem-se em um bem dos mais valiosos, sendo assim, a segurança e a integridade desses dados são essenciais para quem os manipula.

Prover segurança é garantir que dados sigilosos sejam manipulados apenas por pessoas autorizadas, e dentro de um controle pré-definido. A manutenção dos objetos do BD, é tarefa constante em todo o ciclo de vida do BD, como consequência a monitoração de desempenho também deve ser permanente, com a finalidade de manter o bom funcionamento do BD.

Existem vários níveis de segurança que devem ser considerados, para a obtenção de um grau aceitável de proteção aos dados uma vez que é impossível conseguir segurança total.

Em geral, a equipe que gerencía o BD, trabalha em conjunto com a equipe de desenvolvimento e os vários departamentos da organização, a fim de decidirem as estratégias de segurança, ou seja, definir quais as pessoas autorizadas a manipular os dados do BD.

Os usuários podem manipular esses dados de duas formas: através de programas de aplicação, desenvolvidos pela equipe de desenvolvimento ou através de linguagens interativas, que acompanham os SGBDs.

- As linguagens interativas oferecem um conjunto de instruções, que podem ser manipuladas de acordo com as necessidades dos usuários. A Linguagem SQL (*Structured Query Language*) disponível para a maioria de BDs relacionais é um exemplo [Date 91]. Essas linguagens permitem ao DBA, diferenciar categorias de usuários, ou seja, usuários com manipulação de leitura, outros de leitura e gravação e assim por diante. A definição dessas categorias, devem ser criteriosas, pois instruções mal utilizadas afetam diretamente os dados do BD .

- Por outro lado, os programas de aplicação são codificados, utilizando uma linguagem de programação (Cobol, Fortran, C). As instruções de manipulação da linguagem SQL por exemplo, podem ser embutidas nesses programas para que se possa acessar o BD.

O DBA deve especificar a abrangência de acesso aos dados, tanto para os programas de aplicação, quanto para os usuários que utilizam uma linguagem

interativa. Os acessos podem ser atribuídos em níveis ou tipo, como segue: autorização de leitura, inserção, atualização e eliminação de dados.

Os níveis podem ser combinados, para permitir um maior grau de acesso, de acordo com as necessidades de cada usuário ou programa de aplicação.

A linguagem SQL, possui mecanismos para que se possa implementar as estratégias de segurança. Um desses mecanismos consiste das visões, que restringem os acessos de usuários, a pequenas porções de uma ou mais tabelas [Date 91]. Esses mecanismos no entanto, não definem que operações podem ser aplicadas sobre visões ou tabelas. As instruções *grant* e *revoke* complementam essas especificações. A primeira define os tipos de acessos permitidos, a segunda os remove.

Outra medida de segurança que deve ser observada pelo DBA , consiste em codificar rotinas que realizem *backups d*e tabelas. Estas rotinas indispensáveis são para a restauração parcial ou total desses objetos, com uma periodicidade definida, de acordo com a intensidade de atualização sobre as tabelas, ou a pedido do usuário.

2.1.3 Manutenção

Na prática são discutidas duas fases que antecedem a manutenção do BD, que são: a instalação do SGBD e o projeto lógico do banco de dados.

O termo manutenção, refere-se às alterações que um banco de dados sofre, desde sua criação [Atre 80]. A primeira tarefa de um DBA, é especificar os elementos tais como: volume dos discos e espaço de armazenamento, separação física e exigências de afinidades. Esse trabalho de nível interno, é essencial para o bom desempenho do SGBD [Date 91].

A segunda fase está relacionada com o processo de modelagem do BD. O DBA deve ter uma visão geral da organização, para identificar o relacionamento e o fluxo de informação entre os vários departamentos [Atre 80]. Em seguida, com as prioridades da organização, deverá especificar o projeto lógico, onde serão definidas as entidades e seus relacionamentos, que poderão tornar-se tabelas.

E finalmente é feito o projeto físico, onde é especificado como os dados serão representados internamente. Os principais serviços que podem ser executados nesta fase são: criação, alteração e remoção de tabelas, índices e *views*; carga de dados em tabelas do BD. Essas operações geralmente são executadas, através de uma linguagem de definição de dados (DDL). A linguagem SQL, oferece um conjunto de instruções que permite, o mapeamento do esquema conceitual para o esquema interno. As instruções de mapeamento dessa linguagem (DDL) são: *create table, create index, create view, drop table e drop index* [Date 91].

As fases um e dois, mostram que o DBA deve conhecer com uma certa profundidade, o equipamento de hardware e o sistema operacional onde será instalado o SGBG. E, principalmente, o funcionamento do gerenciador, desde a sintaxe dos comandos que manipulam o BD, até sua arquitetura, de modo a minimizar os problemas de desempenho, que serão discutidos da próxima secção.

2.1.4 Desempenho

O desempenho de um BD, está intrinsicamente ligado às definições dos objetos que o compõem, e de que forma eles serão manipulados.

Alguns fatores externos ao BD, podem causar um mau funcionamento do gerenciador, tais como: sobrecarga do sistema, problemas de comunicação, capacidade de armazenamento insuficiente etc [IBM 88].

Os dados de uma tabela podem ser manipulados por um programa de aplicação, ou por uma linguagem interativa, como foi descrito na seção 2.1.2.

As operações que mais sobrecarregam um SGBD, são aquelas que precisam fazer uma leitura sequencial em tabelas com muitas linhas, pois requerem constantes acessos a disco (E/S).

O DBA deve, portanto, interagir constantemente com a equipe de desenvolvimento, para conhecer suas necessidades de dados sobre o BD, e fornecer soluções adequadas a execução dos programas aplicativos [Atre 80].

Uma solução para o problema acima, seria definir uma área de página de *buffer*, com grandes proporções.

As operações que necessitarem manipular apenas uma linha de uma tabela, podem ser ineficientes, caso um índice não seja especificado para acelerar a recuperação da mesma. O DBA então deve especificar pelo menos um índice para a tabela [Date 91]. Esse índices podem tornar-se fragmentados, através de frequentes operações de inserção e remoção de linhas da tabela, implicando em queda de desempenho, portanto, o DBA deve monitorar constantemente o BD, e reorganizar as tabelas com índices fragmentados.

A monitoração de desempenho, certamente é um desafio aos DBAs, pois várias razões podem levar à queda de desempenho. As ferramentas de monitoração que acompanham os gerenciadores, muitas vezes são insuficientes ou não oferecem clareza nos seus resultados.

Um exemplo de um recurso bastante útil são os catálogos de um SGBD. Eles são uma espécie de dicionário de dados, armazenando informações sobre tabelas, aplicações e índices, que permitem ao DBA, acompanhar o desempenho do BD.

2.2 Infra-estrutura de Comunicação

O DBA-CS foi idealizado de forma a permitir a gerência de BDs em qualquer parte da rede. Esta rede pode ser local, a longa distância, heterogênea ou formada pela integração de várias outras redes.

Nas próximas seções serão discutidos, os aspectos conceituais para a implementação deste projeto. Primeiramente é apresentado o conjunto de protocolos TCP/IP, utilizado neste projeto como base de comunicação entre máquinas possivelmente heterogêneas; o modelo conceitual Cliente-Servidor e finalmente o modelo de chamada de procedimento remoto (RPC).

2.2.1 TCP/IP uma Arquitetura de Comunicação

O TCP/IP (*Transmission Control Protocol/Internet Protocol*) é um conjunto de protocolos, que permite que dados sejam transportados entre computadores com arquiteturas diferentes [Commer 91]. Sua grande aceitação no mercado, o tornou um padrão ***de facto.***

A arquitetura TCP/IP é composta por uma arquitetura em camadas, que pode ser comparada com três camadas do modelo OSI (*Open System Interconnection*): camada física, de rede e de aplicação, como mostra a Figura 2.1 [Tanenbaum 89]. O TCP/IP não especifica a camada física, portanto o usuário pode usar qualquer

camada física de transmissão, tanto para **LAN** (*Local Area Network*) como para **WAN** (*Wide Area Network*) [IBM 92].

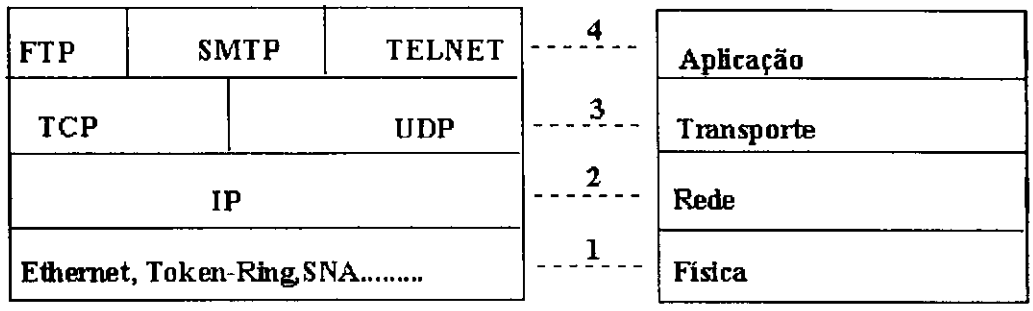

Figura 2.1 - Arquitetura TCP/IP x OSI

A camada de rede, IP (*Internet Protocol*), interconecta uma ou mais redes, ou seja, esta camada providencia o roteamento de mensagens de um dispositivo para outro, em um ambiente local ou sob uma rede a longa distância. Do ponto de vista do usuário, a parte visível do IP são os endereços que cada máquina participante da rede recebe. Tais endereços são únicos, e são chamados **endereços IP** [Moura 92].

O IP define um endereço *string* de 32 bits, para cada *HOST* da rede. Este endereço é constituído de dois campos: o identificador de rede (**NetID**) e o HOST equivalente da rede (**HOST ID**). O NetID é o mesmo para todos os HOSTS de uma determinada rede.

Por exemplo, o endereço 150.165.2.1 é assim descrito: O NetID é representado pelo número 150, enquanto que 165.2.1 identifica o HOST ID.

Apesar desses endereços fornecerem uma representação compacta e conveniente para especificar as máquinas as quais desejamos contactar, usuários preferem usar nomes (mnemônicos). No exemplo anterior o endereço 150.165.2.1 poderia ser codificado como magali.dsc.ufpb.br, que simplificaria o reconhecimento da rede e da máquina, por parte do usuário. A escolha de nomes deve ser única, para evitar ambiguidades.

A terceira camada oferece dois protocolos que transportam informações, entre nós da rede. O TCP (*Transmission Control Protocol*), orientado para conexão,

garante a transmissão ordenada e sem erro entre pares de processos em uma rede, é um protocolo confiável. O UDP (*User Datagrama Protocol*), é um protocolo sem conexão e sem confiabilidade, ou seja, garantia de entrega, mas obtém um bom desempenho, por não sofrer atraso inicial para o estabelecimento de conexão.

Na quarta camada estão os protocolos de alto nível, chamados de protocolo de aplicação, tendo como função básica, efetuar a interface com as aplicações que desejam utilizar os meios de comunicação de dados. Os protocolos primários do TCP/IP são TELNET, FTP e SMTP [IBM 92]. Outros protocolos poderão estar disponíveis em outras implementações TCP/IP [IBM 92].

2.2.2 Modelo Cliente-Servidor

Nesta seção será apresentado o modelo conceitual Cliente-Servidor, que foi utilizado para modelar o DBA-CS, e por conseguinte os tipos de servidores.

Como foi descrito na seção 2.2.1, uma das características do TCP/IP, é permitir que dois programas de aplicação estabeleçam uma conexão entre si, e transportem dados de um para o outro.

Apesar das características descritas anteriormente, o TCP/IP não especifica as regras de interação entre essa aplicações. O modelo Cliente-Servidor (C-S) serve de base para esse tipo de comunicação, especificando que um das aplicações espere indefinidamente, o contato da outra.

O modelo C-S especifica que dois processos distintos sejam cooperativos. O servidor é um processo que oferece serviços aos processos clientes [Commer 93].O servidor recebe um pedido de serviço, executa e retorna o resultado ao cliente. A Figura 2.2 mostra como funciona este esquema.

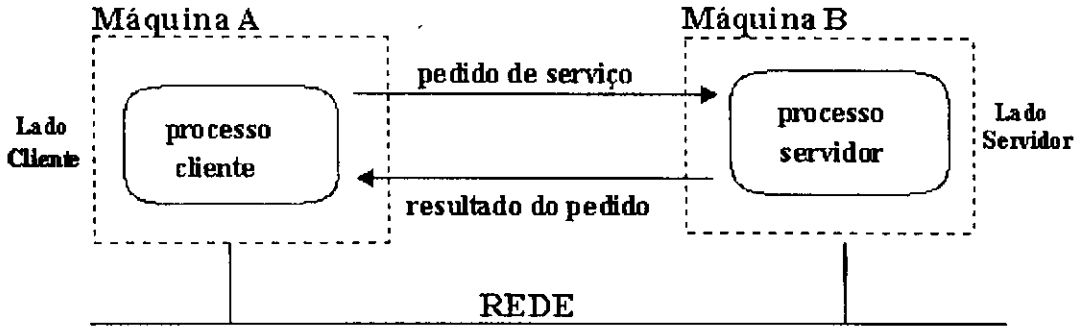

Figura 2.2 - Modelo Cliente-Servidor

Um dos aspectos importantes em uma comunicação C-S, esta relacionado com o tratamento oferecido pelo servidor, para pedidos de serviços por diversos usuários ao mesmo tempo. Eles são classificados quanto à estratégia de serviço; Servidor Iterativo e Servidor Concorrente; e quanto à manutenção de estado: Servidor com estado ou servidor sem estado [Filho 93].

- **Servidores Iterativos e Concorrentes**

Os servidores iterativos executam um único pedido por vez, ou seja, os pedidos são enfileirados e executados na ordem em que chegaram. Esse tipo de servidor é simples de ser implementado, em contra partida, pode causar uma demora considerável para enviar a resposta, dependendo do tipo de operação que se deseje fazer.

Já os servidores concorrentes, possuem a facilidade de atender a vários serviços simultaneamente, criando múltiplos *threads* [1] , um para cada pedido de serviço [Tanenbaum 89]. Esse tipo de servidor executa pedidos simultaneamente, quando a máquina possui múltiplos processadores, caso contrário a concorrência será apenas virtual.

[1] *Thread* representa o agente de controle de uma seqüência de instrucões sendo executado por um programa.

- **Servidores com e sem estado**

O servidor com estado tem como característica principal, manter alguma informação referente aos pedidos dos clientes para atendê-los corretamente [Filho 93]. Essas informações ou estados, são utilizadas pelo servidor, para dar continuidade a novos pedidos, sem que seja necessário enviar um conjunto de informações já armazenadas [Commer 93].

O servidor sem estado não mantém qualquer informação sobre os pedidos de clientes, o estado é mantido pelo próprio cliente, e sua maior vantagem é a tolerância a falhas[Commer 93].

2.2.3 Modelo de Chamada de Procedimento Remoto

Chamada de procedimento remoto, ou RPC (*Remote Procedure Call*), consiste numa biblioteca de funções que permite a comunicação entre processos, desenvolvidos em ambiente distribuído [Corbin 91]. O RPC estende os conceitos de uma chamada local para uma chamada remota, ou seja, funções locais podem ser executadas em outra máquina [Comer 93].

O funcionamento do RPC, está relacionado com o modelo Cliente-Servidor nos seguintes aspectos: procedimentos que serão executados em outras máquinas, são chamados de servidores, e aqueles que fazem a chamada destes procedimentos, são os clientes.

Para que processos remotos possam se comunicar, é necessário que o servidor exista e fique a espera de solicitações. Quando o cliente solicita um serviço, ele envia uma mensagem ao servidor, contendo os parâmetros do procedimento. Neste momento o controle de execução é suspenso e passado ao servidor. Este por sua vez, executa o serviço e retorna uma mensagem contendo a resposta. Ao receber a mensagem, o cliente passa a ser executado após a última instrução RPC. A Figura 2.3 ilustra este procedimento.

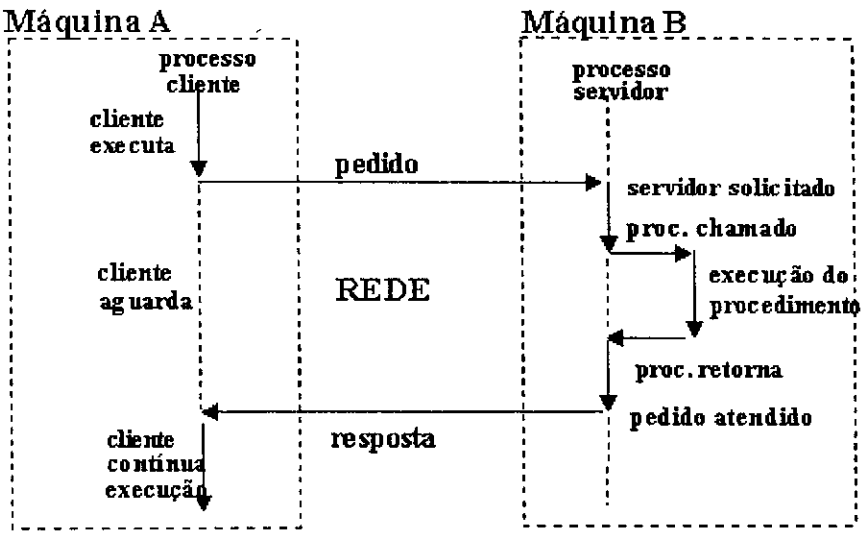

Figura 2.3 - Modelo de Chamada de Procedimento Remoto (RPC)

Algumas particularidades do RPC, tais como: passagem de parâmetros, variáveis globais, protocolo de transporte, entre outras, [Filho 93], devem ser conhecidas pelo programador, para que se obtenha um bom nível de programação.

A administração de banco de dados em geral está restrita a um tipo de SGBD específico, na seção 2.1, procuramos mostrar em linhas gerais, que independente do software que gerenciará o BD, as principais tarefas do dia a dia de um DBA, se resumem em segurança, manutenção e monitoração de desempenho. Foram encontradas algumas dificuldades para estruturar o texto, pois são poucas as referências que abordam especificamente a administração de BDs.

Este capítulo procurou fornecer um embasamento teórico, para situar o leitor nos limites da ferramenta proposta; as principais atividades de um DBA, e uma solução para comunicação de processos distribuídos em máquinas diferentes.

No próximo capítulo, será apresentado a arquitetura do DBA-CS, assim como, os detalhes de sua funcionalidade.

CAPÍTULO III

ARQUITETURA DO SISTEMA DBA-CS

Neste capítulo, será apresentada a arquitetura do DBA-CS, juntamente com a visão geral dos módulos funcionais que compõem este sistema. Ainda neste capítulo, será apresentada a interface do usuário.

3.1 Características Funcionais

O DBA-CS possui os seguintes módulos assim distribuídos: Módulo Cliente e Módulo Servidor, conforme mostra a Figura 3.1.

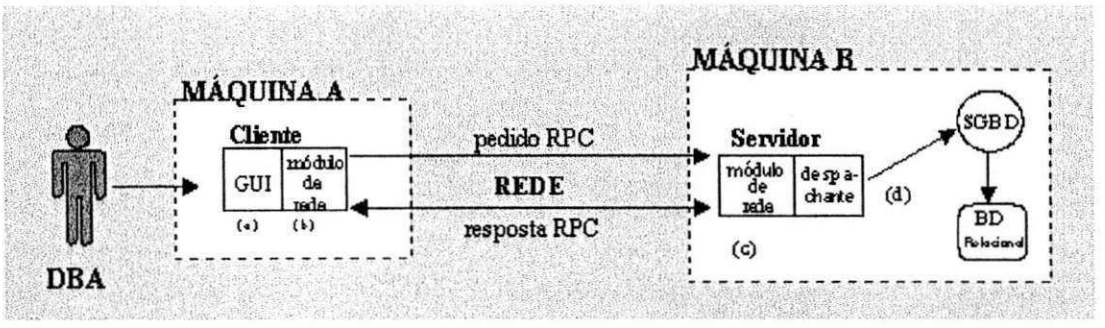

Figura 3.1 *Visão global dos componentes de software do DBA-CS*

O componente cliente está dividido, basicamente, em duas partes. A primeira, é a interface gráfica do usuário (GUI), descrita na Seção 3.4, que permite ao DBA interagir com o DBA-CS (Fig. 3.1-a). A segunda, é um módulo que contém a funcionalidade de rede necessária para executar as operações remotas especificadas

através da GUI (Fig. 3.1-b). Esse módulo envia, através da rede, os pedidos de serviço a serem executados na máquina do servidor.

O componente servidor também está dividido em duas partes. A primeira é um módulo com funcionalidade de rede para receber os pedidos de serviço do cliente (Fig. 3.1-c). A segunda é um módulo de despacho, descrito na seção 3.3, que é responsável pela interação do servidor com o SGDB que vai executar o serviço solicitado pelo cliente (Fig. 3.1-d). Uma vez que o serviço tenha sido executado, o servidor retorna para o cliente o resultado das operações efetuadas pelo SGBD.

A seguir passamos a descrever cada um destes módulos.

3.2 Módulo Servidor

Nesta seção será apresentada a funcionalidade dos componentes que formam o módulo servidor: módulo de rede e despachante, seguido das estruturas de dados que por ele são manipuladas.

O módulo de rede do servidor (Fig. 3.1-c), é responsável pelo recebimento dos parâmetros enviados pelo cliente e por identificar o procedimento remoto a ser executado, retornando o resultado após o término da execução do serviço.

O módulo despachante (Fig. 3.1-d) contém os procedimentos remotos, que farão os acessos ao banco de dados.

O servidor que é executado em *back-end*, fica em execução constante esperando a solicitação de serviços.

Os principais serviços oferecidos por este módulo são descritos abaixo.

- identificar a existência de uma tabela no banco de dados;
- cadastrar usuário no banco de dados (BD);
- autorizar acesso de usuários em tabelas do BD;
- recuperar tipos de acessos de usuários;
- remover acesso de usuários em tabelas:

- copiar dados de tabelas em fita ou disco;
- criar tabelas;
- alterar tabelas;
- deletar tabelas;
- carregar dados em tabelas;
- criar índices para tabelas;
- alterar índices de tabelas;
- recuperar a definição de índices para uma tabela;
- deletar índices de tabelas;
- selecionar tabelas com índices fragmentados;
- selecionar aplicações que acessam tabelas fragmentadas;
- selecionar indicadores de desempenho;

3.2.1 Estrutura de dados do módulo servidor

Nesta seção serão descritas as estruturas de dados (ED's) que este módulo manipula.

Lista de privilégios em tabelas

- **ap_lista_tab:** apontador para uma lista de dados que armazena as tabelas que o usuário tem acesso, e seus privilégios. O nó da lista é composto pelas seguintes variáveis:

  - nome_tabela: nome de tabela;
  - privilégio: *array* que armazena o tipo de privilégio;
  - rem_tipo: *array* que receberá o tipo a ser removido;
  - ap_prox: aponta para o próximo nó da lista;

Lista de tabelas e índices

- **ap_tabelas:** apontador para uma lista, que contém as definições de uma tabela, que serão manuseadas pelas rotinas de criação, alteração e deleção de tabelas. Cada nó da lista é composto pelas seguintes variáveis, assim descritas:

  - tabela : *string* que armazena o nome da tabela;
  - dbspaces: tamanho de cada *dbspace*, que representa o tamanho da área lógica onde serão armazenadas as estruturas do BD.
  - pool: variável numérica que armazena o número do local físico onde serão armazenadas as tabelas e índices.
  - **ap_colunas:** lista que contém as colunas que formarão uma tabela.

  A seguir, descreveremos as variáveis que compõem um nó da lista.
    - coluna: *string* que armazena o nome da coluna;
    - tipo: variável alfa-numérica que armazena o tipo de dados de cada coluna;
    - pnulos: variável do tipo caracter que identifica se uma coluna permite ou não valores nulos;
    - ap_prox: apontador para a próxima coluna;
    - ap_ant: apontador para a coluna anterior;

  - **ap_ind_tab:** lista que fornece o nome dos índices de uma tabela. As variáveis que formam um nó são:
    - nome_ind: *string* que armazena o nome do índice;
    - tipo_ind: variável caracter, que identifica o tipo de índice, único ou não único;
    - status: variável caracter, que identifica o estado atual do índice, se esta fragmentado ou não;
    - ap_prox: aponta para o próximo índice;
    - ap_ant: aponta para o índice anterior;

Lista de áreas lógicas disponíveis no BD

- **ap_dbspaces:** lista que será recebida pelo módulo manutenção, enviada pelo módulo servidor. As variáveis que comporão o nó da lista são:
  - dbspace: tamanho lógico da área, onde as tabelas serão definidas;
    - ap_prox: aponta para o próximo dbspace;
    - ap_ant: aponta para o dbspace anterior;

Lista de definição de índices

- **ap_índice:** apontador para uma lista que contém as definições de índices. Cada nó da lista, é composto pelas seguintes variáveis:
  - tabela: *string* que armazena o nome da tabela, onde foi criado o índice;
  - índice: *string* que aramazena o nome do índice;
  - tipo_ind: variável caracter que identifica, o tipo de estrutura do índice, ou seja, se é único ou não único;
  - **ap_col_ind:** lista que armazena os nomes das colunas que compõem o índice;
    - coluna_ind: *string* que armazena o nome das colunas que formam o índice;
    - ap_prox: aponta para a próxíma coluna;
    - ap_ant: aponta para a coluna anterior;

Lista de estatísticas de desempenho

- **ap_info:** apontador para uma lista de dados que armazena o nome de tabelas manipuladas por uma aplicação, juntamente com outras informações referentes a esta tabela. O nó da lista é composto pelas seguintes variáveis:
    - tabela: *string* que armazena o nome da tabela;
    - pool: variável numérica que identifica a área física onde a tabela está armazenada;
    - criador: *string* que contém o nome do usuário que criou a tabela no BD;
    - **ap_lista_ind:** lista contendo informações sobre índices, definidos para a tabela. As variáveis que formam cada nó são:
        - nome_indice: *string* que armazena o nome do índice;
        - col_ind: *string* que armazena as colunas que compõem o índice;
        - status: identifica o estado do índice;
        - ap_prox: aponta para o próximo índice da tabela;
        - ap_ant: aponta para o índice anterior;

- **vet_fisico:** *array* numérico que armazena o percentual de uso, das áreas físicas do BD.
- **vet_indica:** *array* numérico que aramazena os indicadores de performance do BD;

A figura 3.2, ilustra a estrutura de dados desde módulo

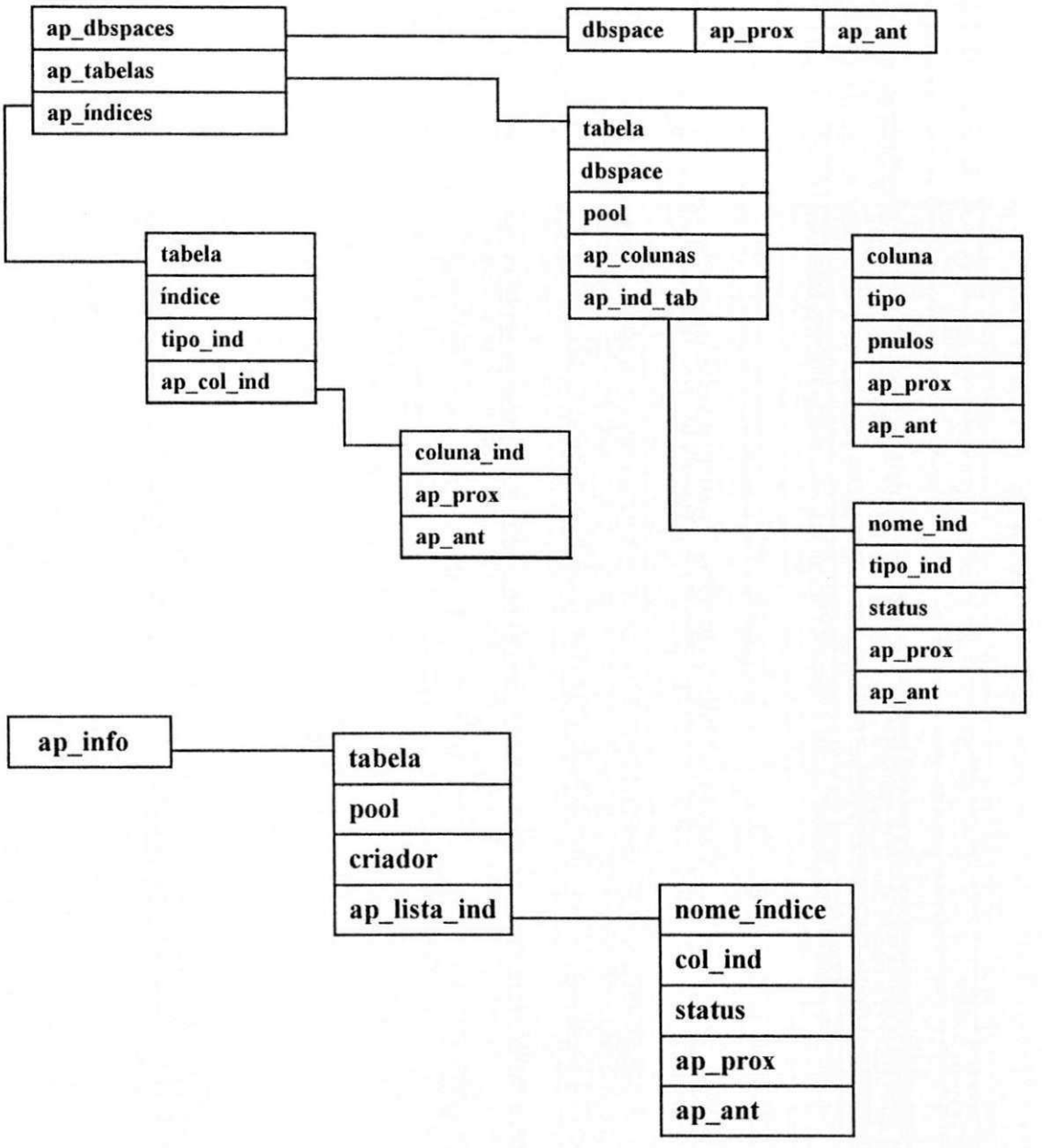

Figura 3.2 Estrutura de dados do módulo servidor

3.3 Módulo Cliente

Este módulo tem como objetivo principal, a comunicação dos processos clientes com os processos servidores, distribuídos em duas máquinas de arquiteturas diferentes, conectadas por uma rede de comunicação.

Este módulo é formado pela GUI (Interface Gráfica do Usuário) e pelo módulo de rede, como foi mostrado na Figura 3.1.

A GUI é responsável pela interação do DBA com o DBA-CS, e será desenvolvida, utilizando um toolkit fornecido pela SUN, denominado Xview [O'Reilly 91], que é executado sob o X windows. Os módulos funcionais da GUI: **Segurança, Manutenção e Desempenho**, serão descrito nas próximas seções.

Quando é feita uma chamada RPC, o módulo de rede do cliente, empacota as estrutura de dados, juntamente com o nome do procedimento remoto e o nome do host, ou seu endereço IP , e envia através da rede para o módulo servidor. Este por sua vez, como foi descrito na seção 3.1.1, desempacota a mensagem, recebe os argumentos, identifica o serviço solicitado, executa e retorna a resposta ao módulo cliente.

3.3.1 Módulo Segurança

Como já foi dito, uma das principais funcões de um DBA é prover segurança [Date 91]. O DBA é responsável por manter cópias (*backups*) do banco de dados, e permitir que apenas usuários autorizados tenham acesso a esses dados. Este módulo tem como objetivo, fornecer funções que controlem o acesso ao BD, e que realizem cópias de tabelas, permitindo que elas sejam recuperadas, caso alguma eventualidade ocorra, seja por falha humana ou do *hardware*.

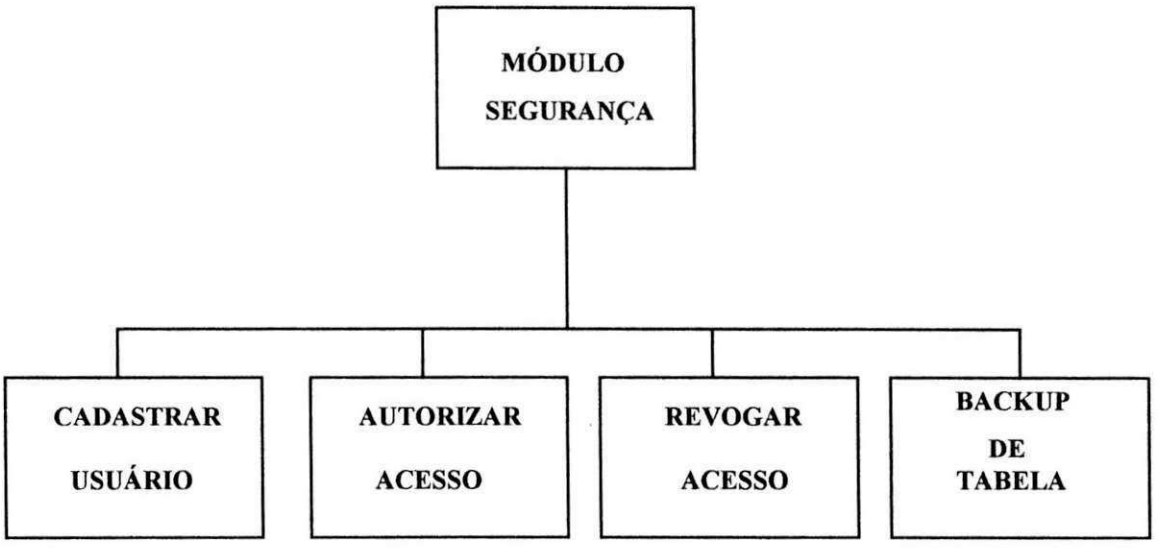

Figura 3.3 - Módulo Segurança

Nas próximas seções serão discutidas cada uma destas funções.

Cadastrar Usuário

Esta função permite o cadastramento de usuários. Para que possam ter acesso ao BD, o DBA deve inicialmente fornecer o nome e a senha desse usuário, e em seguida selecionar o tipo de usuário dentre os seguintes:

- **DBA;**
- **RESOURCE;**
- **CONNECT;**

O usuário DBA como administrador do banco de dados, recebe o privilégio máximo que pode ser dado a um usuário. Aqueles cadastrados com o privilégio de Resource, estarão autorizados a manipular tabelas públicas e criar suas próprias tabelas, os usuários Connect têm acesso apenas às tabelas públicas.

O módulo GUI é responsável pela entrada dessas informações, fazendo todo o tratamento de entradas inválidas e envio de mensagens de erro para o usuário. A descrição da interface é apresentada em detalhes na seção 3.2.

Autorizar Acesso

Esta função é responsável por autorizar o acesso de usuários ou aplicações em tabelas existentes no BD. Quando é fornecido através da interface o nome do usuário ou aplicação , e o nome da tabela, , a função verifica se o usuário é cadastrado e se a tabela existe no BD. Em seguida, a interface fornece a lista abaixo com os privilégios que podem ser atribuídos ao usuário.

- **SELECT;**
- **INSERT;**
- **UPADTE;**
- **DELETE;**
- **ALL;**

O privilégio de *Select* permite que usuários façam apenas consultas nas tabelas, o de *Insert* o usuário poderá fazer inclusões de linhas/registros, o de *Update*

ele poderá fazer modificações nas linhas, o de *Delete*, pode remover linhas e finalmente ALL, obtém todos os privilégios descritos anteriormente ao mesmo tempo.

A seguir o DBA deve especificar o alcance da autorização, ou seja, se o usuário pode ou não repassar a autorização recebida para outros usuários, o *default* do sistema é não repassa. A seleção é feita através de uma lista de escolhas.

Revogar Acesso

Esta função permite que acessos de usuários ou aplicações a tabelas do BD, possam ser revogados, ou seja, removidos. A função segue os mesmos princípios da função de autorizar acesso, o DBA deve fornecer o nome do usuário ou aplicação e especificar quais os tipos de acesso que deverão ser revogados.

A função fará a operação inversa da autorização de usuário. Ou seja todos os privilégios anteriores, ou parte deles serão devidamente removidos. A entrada dessas informações também é feita através da interface, descrita na seção 3.2.

***Backup* de Tabelas**

Esta função tem como objetivo, fornecer ao DBA condições para que seja feito cópia de tabelas do BD. A interface passa o nome da tabela a ser copiada, para a rotina que verifica a existência da mesma no BD. Uma lista contendo todas as colunas desta tabela, é apresentada e torna-se disponível para que as colunas sejam selecionadas em caso de cópia parcial. O escopo do *backup* é o seguinte:

- Especificar se todas as colunas da tabela, ou apenas parte dela serão copiadas;
- Fazer a seleção das colunas no caso da cópia ser parcial;
- Especificar o dispositivo de armazenamento. O *default* do sistema, é o disco.

***Backup* de tabela**

- back_tab: *string* que armazena o nome da tabela;
- ap_col: apontador para uma lista contendo as colunas da tabela;
- coluna: *string* que armazena o nome da coluna. O nó possui uma única variável;

3.3.2 Módulo Manutenção

Uma outra responsabilidade de um administrador de banco de dados (DBA), é decidir a estrutura de armazenamento e a estratégia de acesso aos dados do banco de dados [Date 91].

O objetivo deste módulo, é fornecer funções que permitam a definição dessas estruturas, ou seja, o mapeamento de um modelo conceitual de banco de dados (BD) para o nível interno, através da criação de tabelas e índices.

Estrutura de Dados do Módulo Segurança

O módulo Segurança armazena informações em estrutura de dados, que serão passadas como argumentos para o módulo cliente. Este por sua vez envia para o módulo servidor (através da rede), que executará os pedidos de serviço.

Essas estruturas que compõem cada módulo são descritas a seguir.

Revogar acesso

- rem_usuário: string que representa o nome do usuário, que terá acesso em tabela removido;

Cadastrar Usuários

- cad_usuário: *string* que representa o nome do usuário a ser cadastrado no BD;
- cad_senha: *string* que representa a senha do usuário cadastrado no BD;
- cad_tipo: variável do tipo caracter, que identifica o tipo de usuário cadastrado (D para usuário DBA, R para Resource e C usuário Connect);

Autorizar acesso

- user_tabela: *string* que armazena o nome do usuário a ser manipulado;
- acc_tabela: *string* que armazena o nome da tabela a ser manipulada;
- tipo_autorização: *array* que identificará o tipo de autorização, utilizando a seguinte codificação: o caracter **S** representa privilégio de *Select*, **I** de *insert*, **U** de *update* e **D** para *delete*;
- **alcance:** variável caracter para identificar se o privilégio recebido pode ser repassado para outros usuários. O caracter **S**, significa que repassa e N não repassa.

***Backup* de tabela**

- back_tab: *string* que armazena o nome da tabela;
- ap_col: apontador para uma lista contendo as colunas da tabela;
- coluna: *string* que armazena o nome da coluna. O nó possui uma única variável;

3.3.2 Módulo Manutenção

Uma outra responsabilidade de um administrador de banco de dados (DBA), é decidir a estrutura de armazenamento e a estratégia de acesso aos dados do banco de dados [Date 91].

O objetivo deste módulo, é fornecer funções que permitam a definição dessas estruturas, ou seja, o mapeamento de um modelo conceitual de banco de dados (BD) para o nível interno, através da criação de tabelas e índices.

Foram acrescentadas outras funções, para que manutenções em espaços lógicos e físicos do BD pudessem ser realizadas. A figura abaixo mostra as funções disponíveis neste módulo.

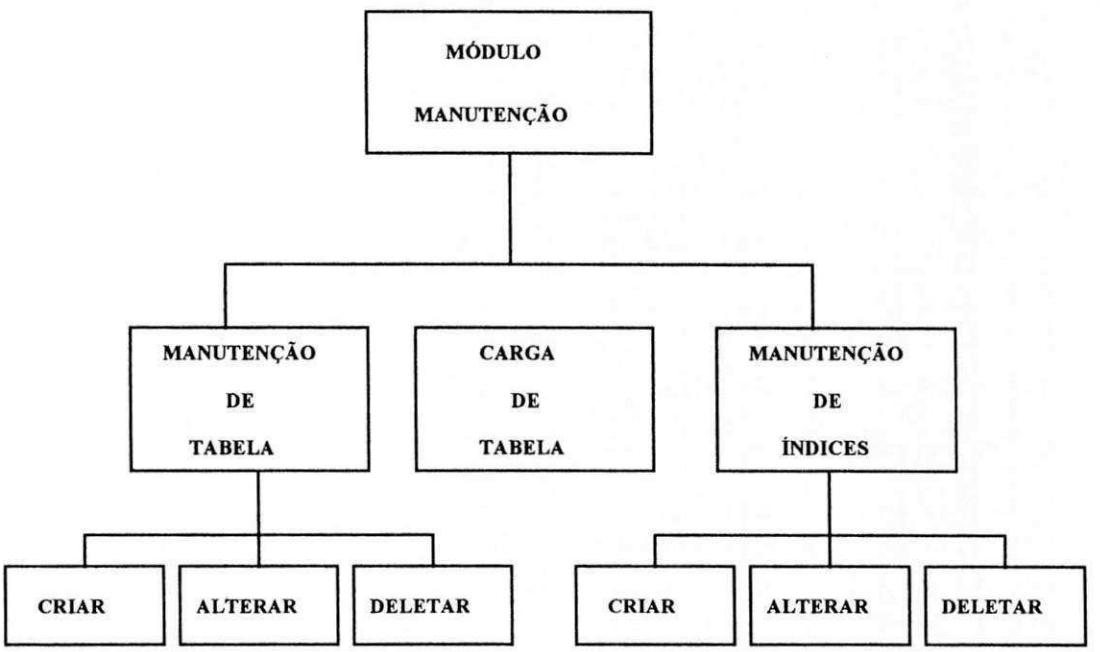

Figura 3.4 - Módulo Manutenção

Manutenção de tabelas

Esta função implementa as operações que permitirão a definição dos dados no BD.

O efeito da operação **CRIAR** é a criação de uma tabela nova e vazia.

O DBA deverá fornecer primeiramente o nome da tabela, para que seja verificada a existência da mesma. Caso já exista, uma mensagem de duplicação de

Índices podem ser removidos do BD, utilizando-se a operação **remover.** A função pede o nome da tabela, manipula o BD e apresenta ao usuário todos os índices pertencentes a esta tabela, em seguida deve ser selecionado o índice a ser removido e a confirmação da operação.

Carga de tabelas

Grandes quantidades de dados podem ser adicionados a tabelas, através da operação **carga**. A interface solicita a entrada do nome da tabela e o dispositivo de armazenamento onde estão armazenados os dados. A rotina verifica se a tabela existe no banco de dados, recebe os argumentos passados pela interface, e executa a operação de *backup*.

Estrutura de Dados do Módulo Manutenção

Este módulo manipula as seguintes estruturas de dados já definidas na seção 3.2.

Lista de tabela e índices;

Lista de definição de índices;

3.3.3 Módulo desempenho

Quando um gerenciador é inicializado, o DBA fornece parâmetros que afetam suas características operacionais. Dependendo da quantidade de entradas no banco de dados (tabelas, usuários, aplicações etc.), e das necessidades de

processamento dessas entradas, esses parâmetros devem ser analisados e se necessário atualizados.

Este módulo permite que o administrador de banco de dados, possa realizar a monitoração de desempenho em todos os objetos que compõem um banco de dados, tais como: tabelas, índices, indicadores de desempenho e áreas físicas onde as tabelas são armazenadas.

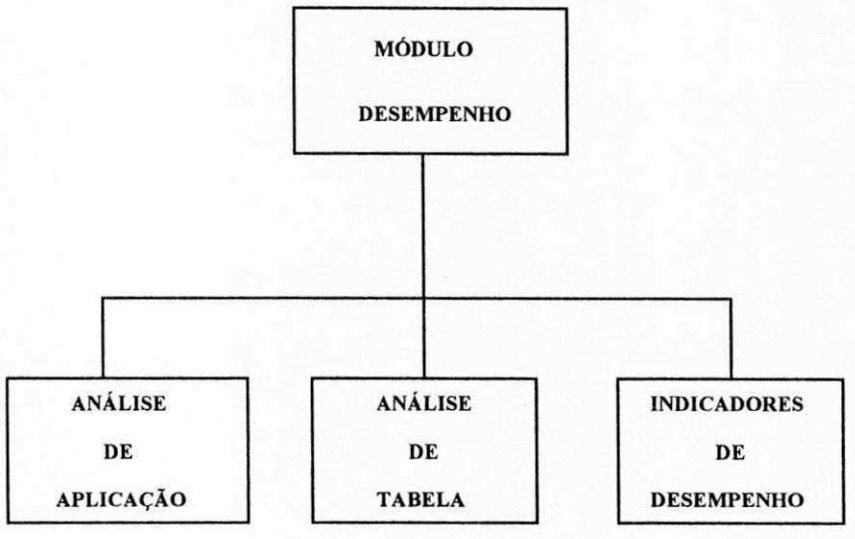

Figura 3.5 - Módulo desempenho

Análise de aplicação

Esta função apresenta através da interface, as possíveis causas de degradação de uma aplicação. Após a entrada do nome da aplicação, a função receberá uma estrutura de dados, contendo informações sobre as tabelas acessadas por esta aplicação. Esses dados serão exibidos pela interface, de forma a permitir uma análise pelo DBA.

O administrador de banco de dados, de posse dessas informações, poderá decidir qual a solução adequada para o problema.

Análise de tabela

Esta função exibe uma lista de todas as tabelas do BD, que estejam com índices fragmentados, devido as operações de inserção, atualização e remoção, e também exibirá aquelas tabelas que estejam sem nenhuma definição de índice, e que certamente provocam degradação.

Indicadores de desempenho

Um SGBD antes de ser executado, ou seja, ficar disponível para operação, requer que alguns parâmetros sejam inicializados. Tais parâmetros são responsáveis pela adequação do ambiente, garantindo boa performance do gerenciador.

Parâmetros tais como: número de usuários conectados ao BD, intervalo para liberação de páginas de dados na memória, especificação de tamanho de buffers e forma de recuperação do BD em caso de falha, devem ser constantemente avaliados de forma a serem atualizados de acordo com as necessidades do gerenciador.

Esta função é responsável pela monitoração desses parâmetros e, quando selecionada, apresenta a situação atual desses valores. A partir dos recursos gráficos da interface, a exibição destes indicadores é feita de modo a destacar os parâmetros mais críticos através do uso de cores e gráficos.

Estrutura de Dados do Módulo Desempenho

As estruturas de dados descritas abaixo, são enviadas do módulo servidor para este módulo (desempenho).

Vet_indica;

Lista de estatística de desempenho;

3.4 Interface com o Usuário

A interface do DBA-CS, foi projetada de forma que o usuário (DBA) ao executar a funcionalidade da ferramenta possa fazer um relação direta com as rotinas de segurança, manutenção e análise de desempenho.

O objetivo maior desta interface, é procurar tornar o trabalho de administração do banco de dados mais eficiente, através do agrupamento das informações sobre diferentes BDs sob uma mesma terminologia e em um único dispositivo.

Em alguns gerenciadores relacionais, por exemplo, o SQL/DS da IBM [IBM 91b], o DBA para fazer uma análise de desempenho, precisa fazer uma série de consultas ao catálogo do BD, e ao mesmo tempo reter mentalmente os resultados anteriores, método improdutivo para uma seqüência de consulta muito longa. O DBA-CS permite que essas operações sejam executada sem um mesmo ambiente e as informações sejam apresentadas em uma única janela.

Um outro objetivo foi tornar transparente para o usuário a administração de BDs em sistemas distribuídos heterogêneos.

3.4.1 Estilo de Interação

O estilo de interação escolhido foi o do *Open-Look* [SUN 90] que consiste de: seleção por menus, auxiliado pelo *mouse*. Esse tipo de estilo, possui várias vantagens, uma delas está no fato de que podem ser eliminados o tempo de treinamento e memorização de uma seqüência complexa de comandos [*Shneiderman* 87].

3.4.2 Características dos Usuários

Os usuários desta ferramenta são os administradores de banco de dados. São usuários experientes, que dominam o equipamento que operam, utilizando com freqüência, as ferramentas disponíveis para o trabalho de gerência.

3.4.3 Níveis de Acesso ao Sistema

Existem dois níveis de acesso para que se possa executar o DBA-CS: No primeiro, o usuário selecionará, qual o domínio que deseja trabalhar, através de uma lista dos domínios onde provavelmente estarão distribuídos os bancos de dados. No segundo nível, devem ser selecionados quais os BDs que se deseja acessar, ver Figura 3.6. Após a passagem por esses dois níveis, a tela principal do DBA-CS indicará a disponibilidade da ferramenta, Figura 3.7.

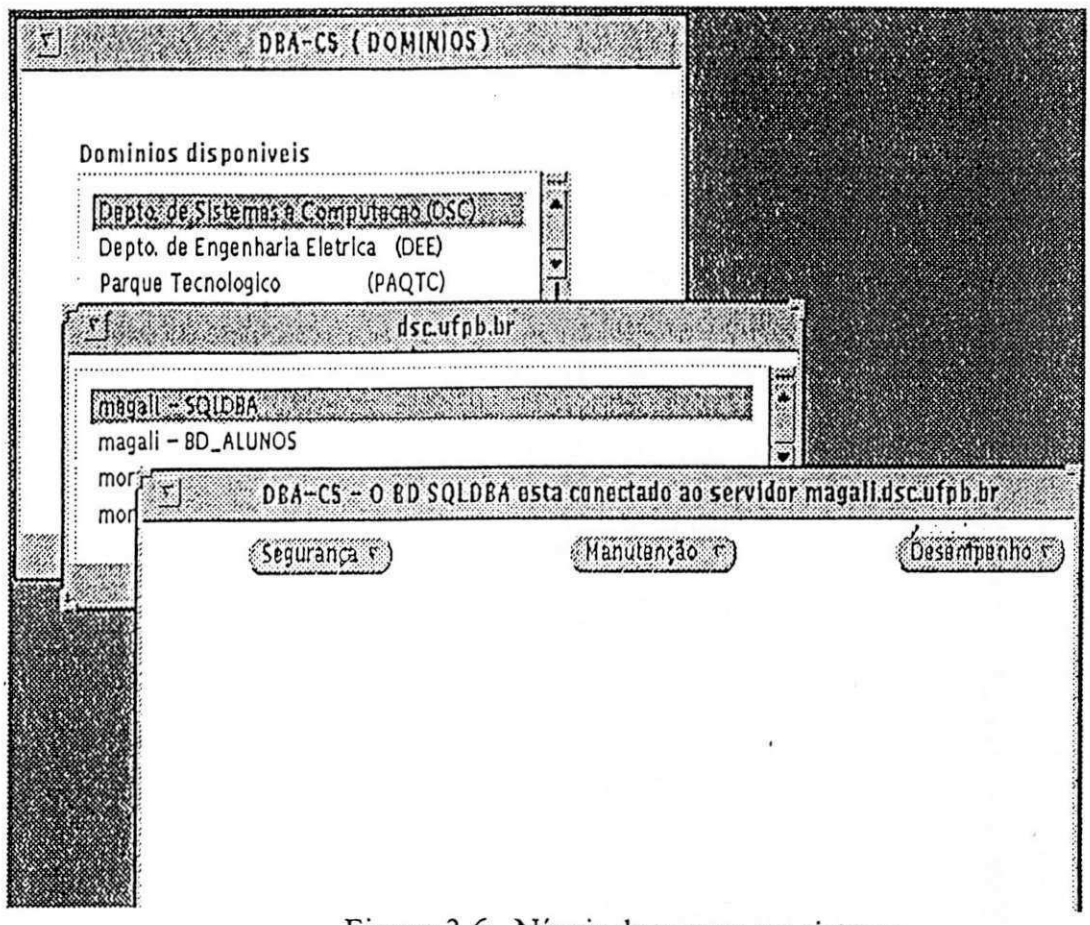

Figura 3.6 Níveis de acesso ao sistema

3.4.4 Recursos oferecidos

Como foi descrito na seção 3.4.1, a GUI do DBA-CS se baseia no padrão *Open-Look* da SUN, utilizando um estilo de interação que poderá facilitar e acelerar o treinamento dos usuários.

A Figura 3.7 mostra a tela principal do DBA-CS. Uma breve apresentação dos seus principais componentes é feita detalhadamente no "Manual do Usuário", o Apêndice A.

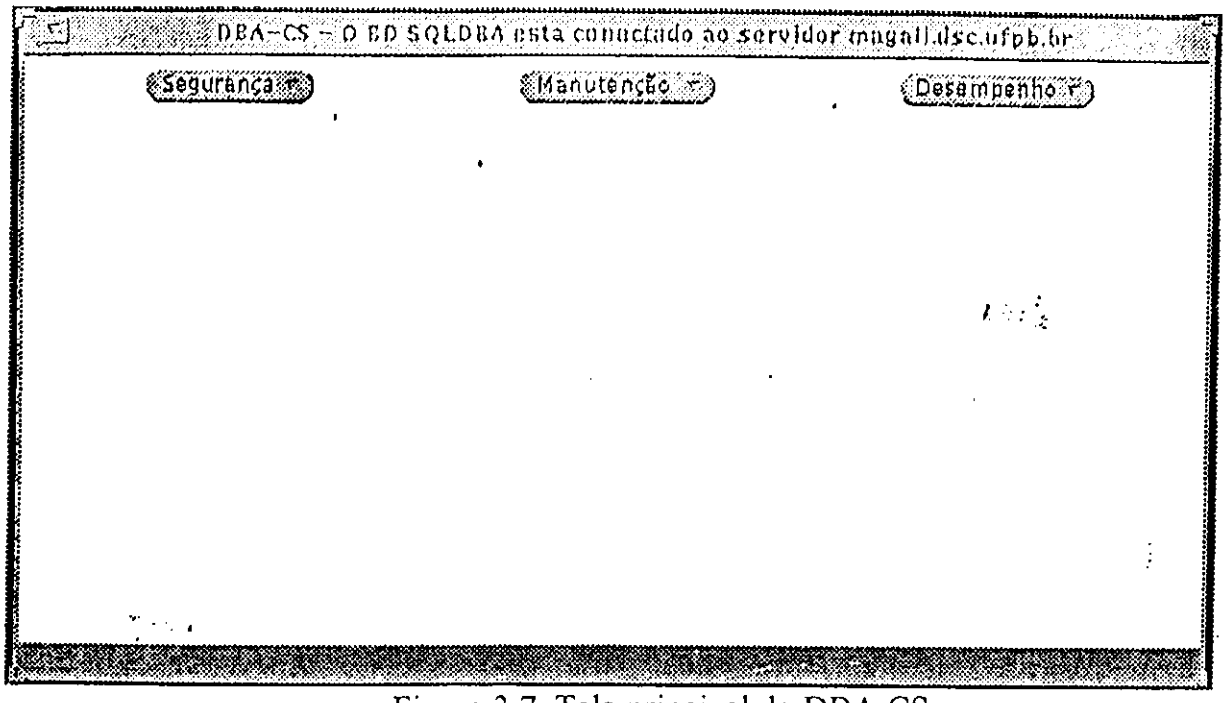

Figura 3.7 Tela principal do DBA-CS

Destacamos na funcionalidade da ferramenta o aspecto de desempenho.

A Figura 3.8 mostra um exemplo de como era apresentado os indicadores de desempenho do BD. O DBA era responsável em fazer alguns cálculos, para identificar um possível problema, por exemplo, para saber se as páginas de *buffer* eram suficientes para as operações que estavam sendo executadas , ele teria que calcular a média entre o parâmetro *Lpagbuff* e *Pageread*. Este trabalho foi minimizado, através do cálculo automático feito pelo DBA-CS, para os parâmetros envolvidos em algum tipo de relação. A Figura 3.9, mostra a nova versão para a função que faz análise dos indicadores de desempenho.

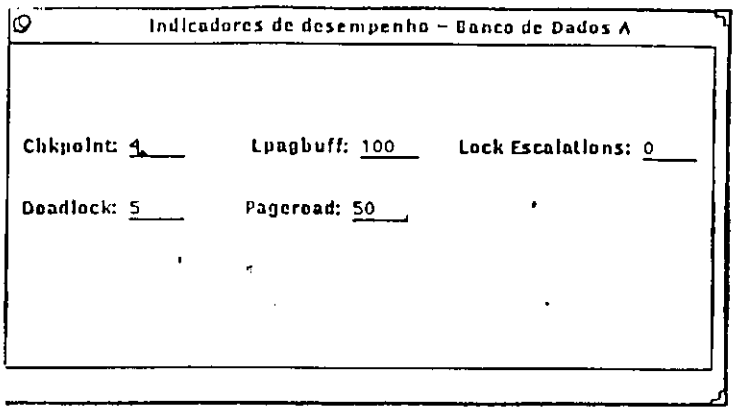

Figura 3.8 Indicadores de desempenho anterior

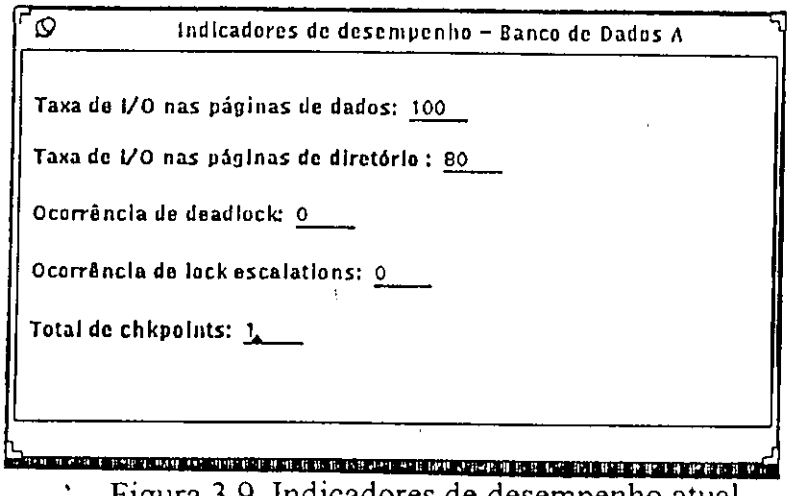

Figura 3.9 Indicadores de desempenho atual

Embora a solução adotada neste trabalho forneça uma representação satisfatória desses indicadores, uma outra alternativa seria apresentar alguns desses parâmetros em uma forma gráfica.

A monitoração de desempenho de alguns desses parâmetros poderia ser diária, onde um gráfico do tipo *pizza* utilizando um código de cores, ajudaria o DBA a identificar mais rapidamente, aspectos tais como; sobrecarga do sistema, podendo assim fazer os ajustes necessários para conseguir melhor desempenho.

O DBA-CS devido às suas características de modularidade (Figura 3.1), permite que seja adaptado para outras plataformas, como o PC por exemplo. A GUI pode ser desenvolvida no padrão *Windows*, que conta com vários *Toolkits* para as versões disponíveis no mercado, e ainda pode contar com o TCP/IP, que já está sendo fornecido por alguns fabricantes, possibilitando uma maior interoperabilidade entre arquiteturas distintas.

CAPÍTULO IV

CONSIDERAÇÕES SOBRE A IMPLEMENTAÇÃO

Neste capítulo são abordadas as considerações feitas na fase de projeto e implementação do DBA-CS. Inicialmente apresentaremos as ferramentas utilizadas na programação do DBA-CS, depois a comunicação entre os processos C-S.. Em seguida, são apresentadas as considerações sobre os requisitos funcionais de rede, para viabilizar a comunicação entre processos distribuídos. Finalmente são descritos os procedimentos de avaliação da ferramenta implementada.

4.1 Desenvolvimento da GUI

A interface gráfica do DBA-CS, foi implementada utilizando XView *Toolkit*, que é baseado no sistema de janelas X Windows. O XView como qualquer outro *toolkit*, fornece um conjunto predefinido de objetos para interface do usuário, tais como: janelas, menus, botões, barras de rolagem etc. A aparência e funcionaliade desses objetos seguem a especificação OPEN LOOK, desenvolvida pela SUN e AT&T, [O'Reilly 91].

Também foi utilizado o Guide (*OpenWindows Developer's*), que se baseia no ambiente de aplicação OpenWindows. Esta ferramenta oferece um ambiente gráfico que facilita a programação visual da interface, através do processo de junção dos objetos fornecidos pelo Xview [O'Reilly 91].

A funcionalidade dos módulos clientes e servidor foi desenvolvida utilizando a linguagem C padrão [Schildt 91].

4.2 Comunicação entre Processos

O DBA-CS foi projetado e implementado, de forma a viabilizar a administração de bancos de dados em plataformas de *hardware* heterogêneo. Para que isto fosse possível, utilizamos o protocolo de comunicação TCP/IP, que permite a comunicação entre computadores distintos, ligados em rede. A rede pode ser local, a longa distância, homogênea, heterogênea ou aindá formada pela integração de várias outras redes (*internet*) [Commer 91].

Toda a comunicação entre o cliente e o servidor foi desenvolvida com *Remote Procedure Calls* (RPC), da Sun Microsystems. O RPC, como foi descrito na seção 2.2.3, é uma API[1] para programação de aplicações em rede que possibilita a comunicação entre processos que residem em espaços de endereçamento de uma ou mais máquinas. O RPC permite ao cliente chamar um procedimento remoto que será executado em outro processo (servidor), e que pode estar em outra máquina [Corbin 91].

Os principais motivos para escolha do RPC da Sun como mecanismo primário de comunicação entre os processos do DBA-CS foram:

- portabilidade: está disponível desde plataformas tipo IBM-PC com DOS/Windows, estações de trabalho Unix até máquinas de grande porte IBM.
- modularidade: ideal para implementação de aplicações cliente-servidor, visto que sua concepção baseou-se neste modelo.
- simplicidade: devido à familiaridade das RPCs com chamadas de procedimento local.
- generalidade: existe um grande número de aplicações convencionais que utilizam RPC, como mecanismo para comunicação entre partes da aplicação.
- eficiência: os esquemas de chamadas de procedimentos são simples o suficiente para possibilitar uma comunicação relativamente rápida.

[1] API (*Application Programming Interface*) são bibliotecas de funções padronizadas, colocadas a disposição do programador para permitir o desenvolvimento de aplicações com um maior nível de abstração, facilitando o trabalho de programação.

4.3 Requisitos Funcionais de Rede

Nesta seção serão apresentados os requisitos funcionais de rede que foram considerados na implementação do DBA-CS: escalabilidade, segurança, tolerância a falhas, desempenho, transparência e transporte.

4.3.1 Transparência

Um importante requisito a ser considerado em projetos de aplicações distribuídas, é a transparência. Do ponto de vista do usuário do DBA-CS, é indesejável que o mesmo lembre da localização dos BDs [Tanenbaum 92].

O conceito de transparência no contexto deste trabalho, esta relacionado com a transparência de localização de bancos de dados, ou seja, como o DBA irá manipular os BDs, independente de sua localização.

O sistema (DBA-CS) não atende esse requisito completamente, pois não existe uma política de um espaço único de nomes para os BDs. O DBA de cada domínio pode escolher o nome do BD que lhe convier, sendo assim existe a possibilidade de haver conflito de nomes.

Apesar de não atender completamente o requisito transparência, O DBA-CS pode alcançar tais objetivos considerando as duas possíveis soluções:

- ### A construção de uma camada sobre o DBA-CS, que utilizaria uma codificação (do tipo: rede+máquina+BD = apelido) para alcancar um maior nível de transparência.
- ### Utilizar o serviço de *naming*, que geralmente está disponível em ambientes de computação distribuída. Este serviço considera a existência de um espaço único de nomes, para todos os objetos da rede.

4.3.2 Segurança

Os aspectos de segurança considerados no projeto e implementação do DBA-CS foram: autorização e auditoria.

Um arquivo de auditoria denominado **LogDba**, monitora todos os eventos que ocorrem no sistema, as operações são armazenadas nesse arquivo, que poderá ser analisado posteriormente pelo DBA.

A ferramenta não especificou um critério de autenticação, mais pode se adaptar ao protocolo Secure RPC, pois ele está integrado com a API de RPC da SUN [Dirffle-76].

4.3.2.1 Autorização

O DBA-CS considerou um esquema de autorização próprio. O DBA de cada domínio pode tornar visível os BD's por ele administrado, basta fornecer os nomes dos bancos de dados, quando for inicializar o servidor, por exemplo, supondo que o nome do servidor seja **server** e que em um determinado domínio existam três BD's, **DB1, DB2 e DB3**, o resultado da seguinte operação: **server DB1 DB2**

Seria a visualização dos BDs DB1 e DB2, por todos os DBA's de todos os domínios, enquanto que DB3 estaria fora desta lista de autorização.

Para o futuro, recomenda-se a utilização de lista de controle de acessos (ACL), onde uma ACL está associada a cada recurso e possui todos os domínios que podem ter acesso ao recurso [Coulouris 88].

4.3.3 Escalabilidade

O DBA-CS foi projetado de forma a ser escalável, do ponto de vista de gerenciamento de BDs, ou seja, ele gerencía tanto um BD, quanto centenas deles, tanto em uma LAN (*Local Area Network*), WAN (*Wide Area Network*), desde que exista interoperabilidade entre as redes envolvidas, com o TCP/IP por exemplo.

O DBA-CS utiliza *broadcast* para a localização dos BDs, no entanto esta solução limitava a escalabilidade do sistema, pois a amplitude do *broadcast* normalmente limita-se a rede local LAN [SUN 88], sendo assim, foi necessário a especificação do servidor (mestre) , para garantir a escalabilidade. Este servidor é o mesmo que faz o despacho dos pedidos SQL para o SGBD.

4.3.4 Desempenho

Considerando que as operações que um DBA realiza sobre um BD não são críticas, no sentido de obter respostas em tempo real, implementamos um servidor interativo, sua simplicidade gera eficiência.

A utilização de servidores correntes, é uma solução para obtenção de um melhor desempenho dos pedidos de clientes, feitos em paralelo. Estes servidores utilizam a API de *threads*, a fim de atender paralelamente as solicitações concorrentes [Tanenbaum 92].

O uso de *caches*, tanto em clientes quanto em servidores, reduzem o tráfego na rede e a carga de trabalho dos servidores. *Caches* em clientes, armazenam os resultados de RPCs anteriores, que poderão ser consultadas a fim de obter resultados desejados, evitando assim, RPCs desnecessárias [Filho 93].

4.3.5 Tolerância a falhas

Algumas operações realizadas no banco de dados são **não idempotentes**, isso significa que essas operações não podem ser realizadas mais de uma vez, por exemplo uma operação que atualiza os catálogos do BD (um Create Table).

Suponhamos que o serviço de criação de tabela seja requerido, o servidor realiza a operação e falha antes de responder ao cliente, como o sistema não faz manutenção de estado, o cliente deve repetir a operação, mas como a tabela já tinha sido criada, um código de erro é retornado, e o cliente tem como saber se tabela foi criada ou não.

Este caso é análogo a dois DBA's trabalhando no mesmo BD simultaneamente, e caso estejam fazendo a mesma operação (por exemplo, tentando criar a mesma tabela), um deles vai receber esse código de erro, em resumo, o próprio SGBD trata esse tipo de ocorrência.

Em relação a falha de máquina, o DBA-CS considera que os servidores são inicializados na hora do *boot*, o administrador da rede coloca os servidores no RC (arquivo de sistema Unix), sendo assim, toda vez que a máquina é re-inicializada, os servidores são colocados em execução automaticamente.

Não foi considerado a situação de queda apenas do servidor, mas pode ser resolvido utilizando um serviço de monitoramento dinâmico, do tipo Inetd, ou seja, se o processo cai, o monitor o recoloca em execução.

4.3.6 Considerações de transporte

Como as operações do DBA-CS são realizadas sobre banco de dados, e uma das características fundamentais desse tipo de software é assegurar que os dados de BD sejam corretos, optamos pelo TCP como protocolo de transporte entre C-S. Este protocolo nos garante maior confiabilidade, garantindo que as informações

transmitidas pela rede, não sejam perdidas e que cheguem a seu destino na mesma ordem que foram enviadas.

Para a comunicação entre servidores, o protocolo para fazer o *broadcast* foi o UDP, nesses casos a confiabilidade é garantida (este caso ocorre quando o sistema inicia a localização dos BD's em qualquer ponto da rede).

4.3.7 Tratamento da Heterogenidade

Em um ambiente de rede heterogêneo, é possível que os parâmetros e resultados das RPCs, feitas entre máquinas com arquitetura diferentes, tenham representação interna de dados incompatíveis, causando problemas na troca de dados entre clientes e servidores. Por exemplo, microcomputadores tipo IBM-PC adotam tabela de caracteres ASCII, enquanto que mainframes IBM adotam EBCDIC. Para evitar esse problema, todas as RPCs feitas pelos clientes e servidores do DBA-CS utilizam o protocolo de apresentação *External Data Representation* (XDR) da Sun [SUN 88]. O XDR está integrado com a biblioteca de funções da API de RPC, o que facilita a programação de RPCs com tratamento de heterogeneidade.

4.4 Testes

Devido a não disponibilidade de um SGBD, a validação do DBA-CS ficou limitada a simulações no que diz respeito à interação entre o servidor e o SGBD. Ou seja, as instruções SQL geradas atuaram sob estruturas de dados, que simulavam os objetos do BD.

A ausência do SGBD não prejudicou os testes, pois as instruções SQL geradas pelo sistema foram constatadas no servidor, apenas os objetos do BD não estavam disponíveis, mas a integração com um SGBD pode ser facilmente resolvida,

pois a linguagem de programação utilizada para implementar o servidor, é considerada uma linguagem hospedeira para o SQL/DS.

Os testes do DBA-CS foram realizados utilizando a RPP (Rede Paraibana de Pesquisa), considerando os seguintes domínios de rede: Departamento de Sistema e Computação - DSC e Departamento de Engenharia Elétrica - DEE. Os dois departamentos, possuem uma rede local Ethernet e executam o sistema operacional SUN OS 4.1.1.

O módulo cliente do DBA-CS foi executado no DSC, enquanto que o módulo servidor foram distribuídos em várias máquinas, algumas no DSC e outras remotamente, no DEE.

A carga do sistema possui um *overhead* inicial, devido as GUIs consumirem muito recurso de hardware/software. O sistema uma vez inicializado, não gera mais esse *overhead* inicial.

O teste consistiu em gerar na GUI as instruções SQL, para os comandos de segurança, manutenção e desempenho e ver estas instruções no servidor, também foram recebidos dados enviados para a GUI através do servidor.

CAPÍTULO V

CONCLUSÕES

5.1 Análise crítica do trabalho e dos resultados

A tarefa de administração de banco de dados geralmente está vinculada a um SGBD específico, pois cada um tem suas particularidades, como por exemplo, a linguagem de manipulação de dados, que mesmo sendo padrão, como a SQL, possui extensões que as diferem. Contudo, os princípios básicos de administração são os mesmos, ou seja, o DBA estará sempre envolvido com os aspectos de segurança, manutenção e monitoração de desempenho.

O DBA-CS considerou como principais as atividades de administração descritas no capítulo II. A ferramenta está limitada a SGBDs relacionais, que utilizam a linguagem SQL para manipulação dos dados. Mesmo com essa padronização, a ferramenta precisará se adaptar às extensões existentes para esse tipo de linguagem, quando os BDs a serem administrados forem de fabricantes diferentes.

A transparência de localização dos BDs, é outra limitação da ferramenta. Nesta primeira versão o DBA deverá especificar o domínio que deseja trabalhar. Esta limitação é devido à falta de uma política de nomes únicos para os BDs existentes na rede, pois existe a possibilidade de haver um DBA para cada domínio.

Acreditamos que o DBA-CS seja viável, por se tratar de uma ferramenta que permite o gerenciamento simultâneo de vários bancos de dados em um mesmo dispositivo, utilizando as facilidades dos sistemas multitarefas, como o UNIX.

A maioria dos produtos BD C-S, ainda utiliza interface de linha de comando, que é indicado para usuários experientes, acostumados com a sintaxe das instruções de manipulação de dados. O DBA-CS fornece uma interface gráfica, que permite uma interação mais amigável entre o DBA e a ferramenta, além de agilizar a tomada

de decisão, através do agrupamento de dados relevantes à decisão, em uma mesma janela. Destaque-se ainda a complexidade decorrente do acúmulo de funções de gerência de BDs sob SGBDs diferentes em sítios remotos, hipótese esta suportada pela implementação proposta. Esta hipótese desafia a eficiência de uma linguagem de comandos convencional.

5.2 Propostas para continuação do trabalho

O DBA-CS é uma ferramenta composta de módulos (Figura 3.1), sendo assim poderá ser adaptado a outras plataformas. Considerando que se tenha como base de comunicação o TCP/IP, e o mecanismo de RPC para comunicação entre os processos. As seguintes sugestões poderão ser aplicadas em novas versões da ferramenta:

- A interface gráfica (GUI), pode ser adaptada a outras plataformas, como o PC por exemplo, que utiliza o padrão Windows para a interface do usuário;
- Adicionar funcionalidades para o módulo de análise de desempenho, que permitam a análise de indicadores através de gráficos;
- Definir uma política de nomes únicos para todos os BDs da rede, para alcançar um maior nível de transparência.
- Adicionar funcionalidade para o módulo de segurança, que permitam a definição de visões
- Generalizar a abrangência da ferramenta além do modelo relacional, permitindo assim que outros modelos como por exemplo; Banco de Dados Orientado a objetos sejam incorporados ao DBA-CS

5.3 Considerações finais

A ferramenta implementada neste trabalho, poderá ser útil para o trabalho de gerência de banco de dados, uma vez que os objetivos especificados foram alcançados.

Os administradores de banco de dados, poderão a partir desta versão do DBA-CS, alcançar uma maior eficiência em suas atividades, pois vão dispor de uma ferramenta que lhe permite gerenciar vários BDs em um mesmo dispositivo, atendendo assim, à demanda imposta por vários usuários do BDs, simultaneamente.

Esperamos que este trabalho tenha dado uma contribuição para os administradores de banco de dados, através de uma ferramenta que procurou atender aos requisitos básicos nas tarefas de administração; segurança, manutenção e análise de desempenho.

BIBLIOGRAFIA

[Date 91]

Date, C.J. Introdução a Sistemas de Bancos de Dados. Campus, Rio de Janeiro. 1991.

[Shneiderman 87]

Shneiderman, Ben. Desingning the User Interface Strategies for Effective Human-Computer Interaction. Addison-Wesley Publishing Company. USA. 1987.

[Atre 80]

Atre, S. Data base: Structured Tecniques for Design, Performance and Management. A Willy-Interscience Publication. USA. 1980.

[Korth 89]

Korth, Robert L. at all. Sistemas de Bancos de Dados. McGraw-Hill. São Paulo. 1989.

[Guezzi 9]

Guezzi, CArlo at all. Fundamentals of Software Engineering. Prentice-Hall International, Inc. New Jersey - USA. 1991.

[Commer 91]

Commer, D.E. & stevens D. L. Internetworking with TCP/IP - Design, Implementation and Internals, volume II. Prentice Hall. 1991.

[Corbin 91]

Corbin, J. R. The Art of Distributed Applications - Programming Tecniques for Remote Procedure Calls. Springer-Verlag. 1991

[Sun 88]

Sun Microsystems. Remote Procedure Calls - Programming Guide. Moutain View, California. 1988.

[Soares 86]

Soares, Luis Fernando Gomes. Redes Locais. Campus. 1986.

[Kruse 91]

Kruse, Robert L. at all. Data Structured and Program design in C. Prentice-Hall. New Jersey.1991.

[Derfler 93]

Derfler, Frank J. Guia de Conectividade. Campus. Rio de Janeiro. 1993.

[Filho 93]

Filho, Gedeon J. S. Considerações para o desenolvimento de Aplicações Distribuídas em Ambientes Heterogêneos. Dissertação de Mestrado. COPIN-DSC / UFPB - Campina Grande - PB, maio 93.

[Sauvé 92]

Sauvé, JAcques P. Computação Distribuída para os anos 90. Transparência do seminário Infotrends'92, São Paulo.

[O'Reilly 91]

O'Reilly & Associates. Xview Programming Manual. USA. 1991

[Moura 92]

Moura, Antão. Uma visão sobre a família de protocolos TCP/IP, Mundo Unix, Março 1992.

[IBM 92]

IBM Corporation International Technical Support Center, TCP/IP Tutorial and Technical Overview, 1992.

[IBM 91a]

IBM Corporation., Database Administration. Canada Ltd. Laboratory, 1991.

[IBM 91b]

IBM Corporation. Managing SQL/DATA SYSTEM. Canada Ltd. Laboratory, 1991.

[IBM 91c]

IBM Corporation. Application Programming. Canada Ltd. Laboratory, 1991.

[IBM 88]

IBM Corporation Diagnosis Guide for VM/System Product and VM/Extend. Architeture Product - Version 2 realease 2, 1988.

[Tortello 93]

Tortello, J. E. N. Bancos de Dados Cliente/Servidor, PC Magazine. Infobook, Rio de Janeiro, 1993.

[Tanenbaum 89]

Tanenbaum, Andrew. Computer Network.Prentice Hall, 1989.

[Hampton 83]

Hampton, David R. Administração comteporânea. McGraw-Hill, 1983.

[Schildt 91]

Schildt, Herbet. C Completo e Total. McGraw-Hill, 1991.

[Commer 93]

Commer, D.E. & stevens D. L. Internetworking with TCP/IP - Client-Server Programming and Aplplications BSD Socket Version, volume III. Prentice Hall. 1993.

[Diffie 76]

Diffie, W. & Hellman, M. E. New Directions in Cryptography. IEEE Transactions on Infomation Theory, pp - 644-654.

[Tanenbaum 92]

Tanenbaum, Andrew. S. Modern Operation Systems, Prentice Hall.1992.

[Coulouris 88]

Couloris, George F. & Dollimore, Jean. Distributed Systems: Concepts and Design, Addison-Wesley. 1988.

MANUAL DO USUÁRIO - SUMÁRIO

APÊNDICE A

MANUAL DO USUÁRIO

Este apêndice mostra como o usuário poderá utilizar o DBA-CS, a partir de uma estação de trabalho SUN SPARC 2, que executa o sistema operacional SUN/OS 2.

O DBA-CS possibilita que o DBA gerencie um ou mais bancos de dados simultaneamente em um mesmo dispositivo, os BDs podem ser locais ou remotos. A ferramenta utiliza o SGBD SQL/DS da IBM, para manipular os dados do banco de dados.

A.1 Inicialização do sistema

O usuário antes de executar o DBA-CS, precisa definir qual o domínio em que vai trabalhar. Para isto, ele deve primeiramente abrir uma sessão em uma estação de trabalho SUN SPARC 2, e em seguida, executar o programa ***domínio.*** A tela que aparecerá para o usuário é mostrada na figura abaixo.

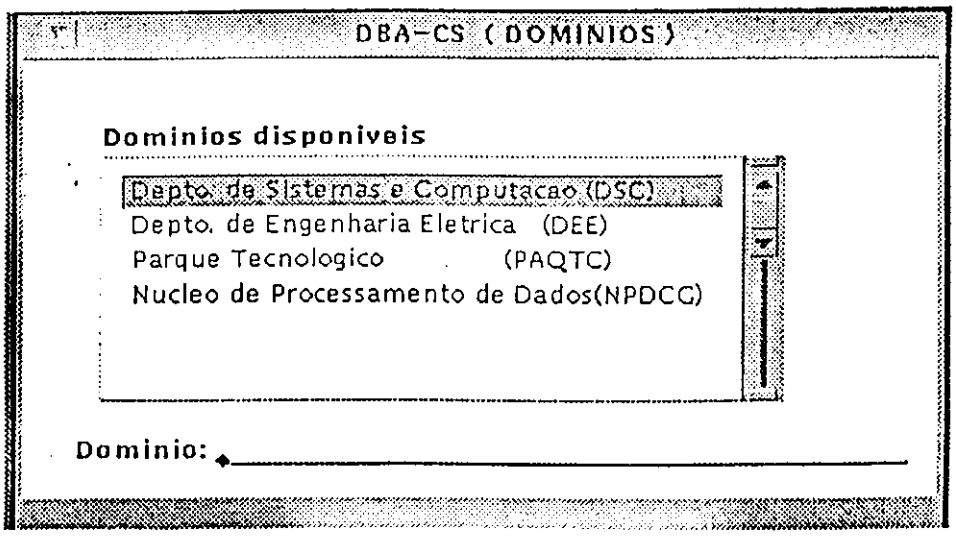

Figura A.1 Lista de domínios

O usuário poderá selecionar um ou mais domínios com ajuda do *mouse*. Para cada domínio selecionado aparecerá uma janela contendo uma lista de banco de dados disponíveis e, na barra de títulos o nome do domínio, como mostra a Figura A.2.

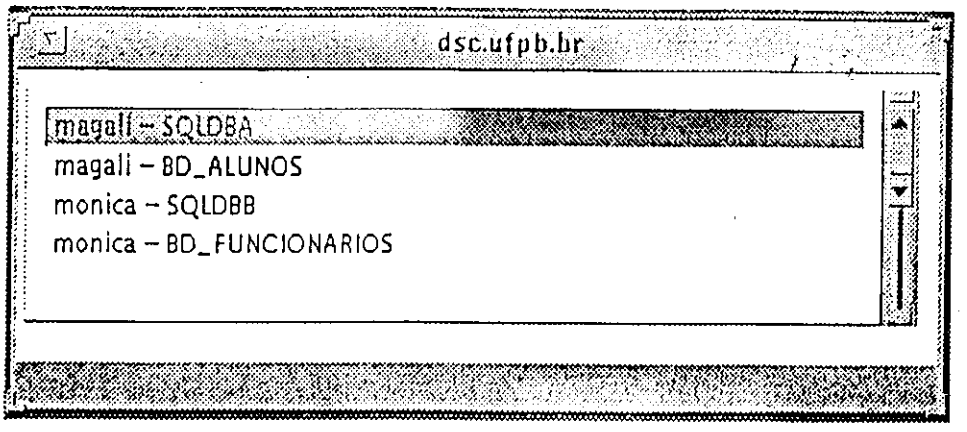

Figura A2 Bancos de dados disponíveis no dominio selecionado

A.2 A tela principal do DBA-CS

Uma vez selecionado o domínio e os BDs a serem gerenciados, aparecerá a tela principal do DBA-CS como mostra a Figura A.3. Cada uma das áreas que compõem a tela principal, é descrita a seguir.

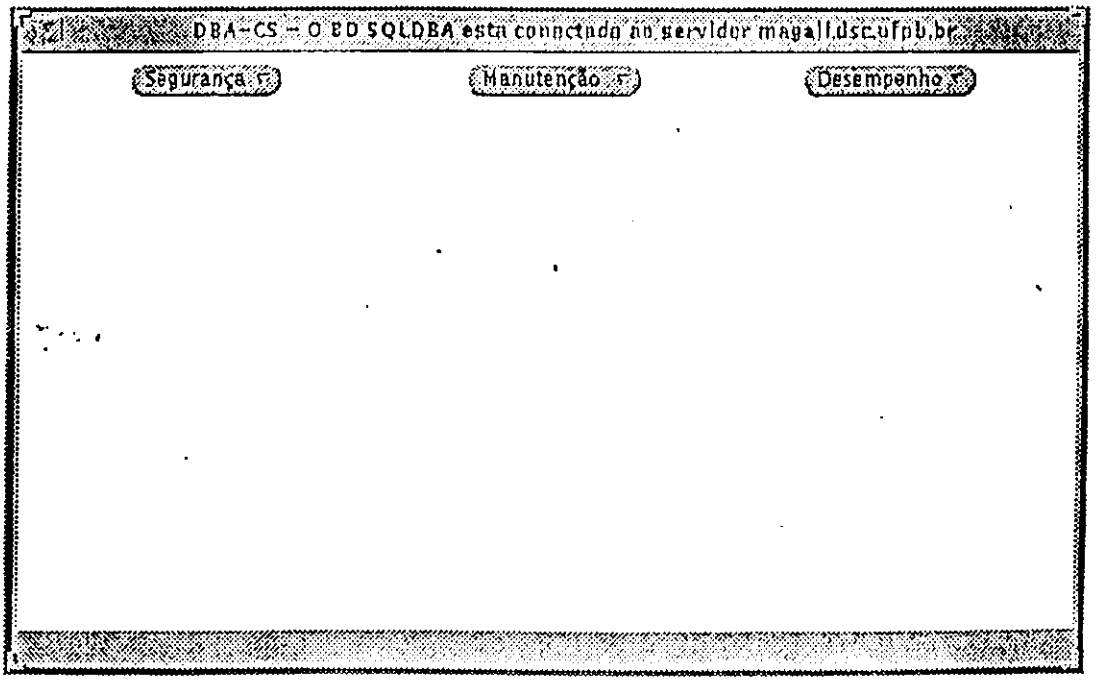

Figura A.3 Tela principal do DBA_CS

- **Barra de títulos**. Contém o nome da aplicação, nome do banco de dados e o domínio a que pertence. Essas informações são úteis para situar o usuário no contexto.

- **Barra de menus**. Os botões: segurança, manutenção e desempenho representam os comandos do DBA-CS.

- **Ícone do menu de controle**. Exibe um menu ***pop-up*** com os comandos de controle da janela: tamanho, posição, encerrar ou iconizar a aplicação, etc..

- **Barra de status**. Mostra mensagens que orientam o usuário do DBA-CS.

A.3 Comandos do menu principal

A.3.1 *Segurança*

Este comando tem como objetivo, fornecer funções que controlem o acesso ao BD, permitindo que apenas usuários autorizados tenham acesso aos dados do BD.e que realizem cópias de tabelas.As operações disponíveis são: cadastrar usuário, autorizar acesso em tabela, remover acessos de uma tabela e fazer cópia de uma tabela, como mostra a Figura A4.

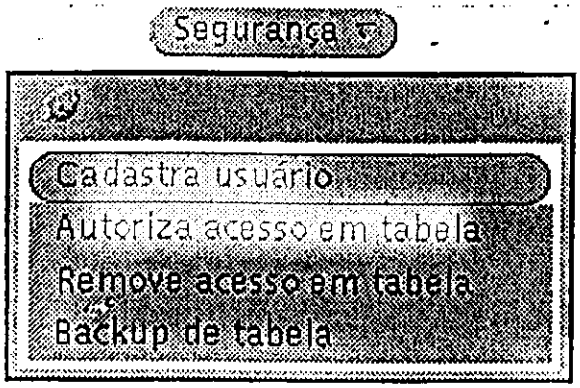

Figura A4 Comandos do botão segurança

A.3.1.1 *Cadastra usuário*

Um usuário para ter acesso ao BD, primeiramente deve ser catalogado. Através desta operação, O DBA poderá especificar o nome do usuário, sua senha de acesso e o tipo de usuário. Nesta ferramenta o usuário pode ser classificado com autoridade de DBA, privilégio máximo, usuário com autoridade de *Resource* ou *Connect*. O tipo de usuário é uma caracteristica do SGBD SQL/DS da IBM, em outros gerenciadores, esta caracteristica pode ter uma outra conotação.

Após o preenchimento dos campos, a operação é confirmada ao se pressionar o botão OK no canto inferior da tela. como mostra figura A5.

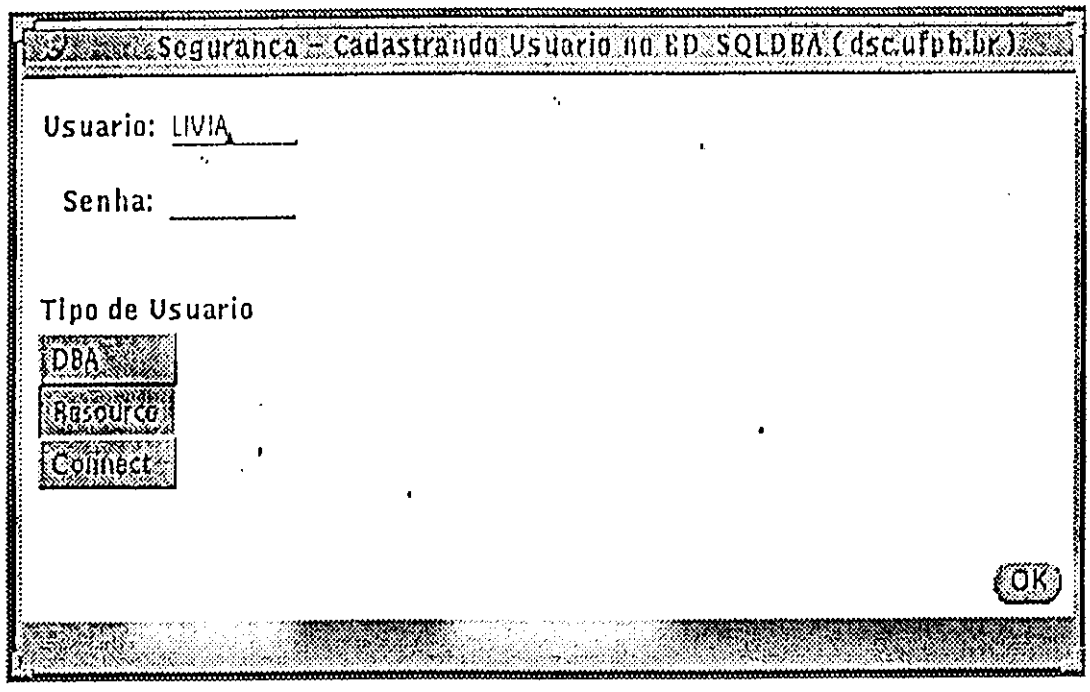

Figura A5 Operação que cadastra um usuário

A.3.1.2 *Autoriza acesso em tabela*

Esta operação permite que seja dado privilégio de acesso em tabelas do BD, para usuários ou aplicações. Deve ser fornecido o nome da tabela e o nome do usuário/aplicação, em seguida os privilégios de acesso podem ser selecionados. É

fornecido através da interface o nome do usuário ou aplicação , e o nome da tabela, a função verifica se o usuário é cadastrado e se a tabela existe no BD. Em seguida deve-se selecionar o tipo de autorização para o usuário. O tipo de autorização pode ser uma combinação entre *select, insert, update e delete*. Deve-se ainda especificar se o usuário poderá ou não repassar o privilégio recebido (o alcance é exclusivo). O botão OK, confirma a operação, como mostra a figura abaixo.

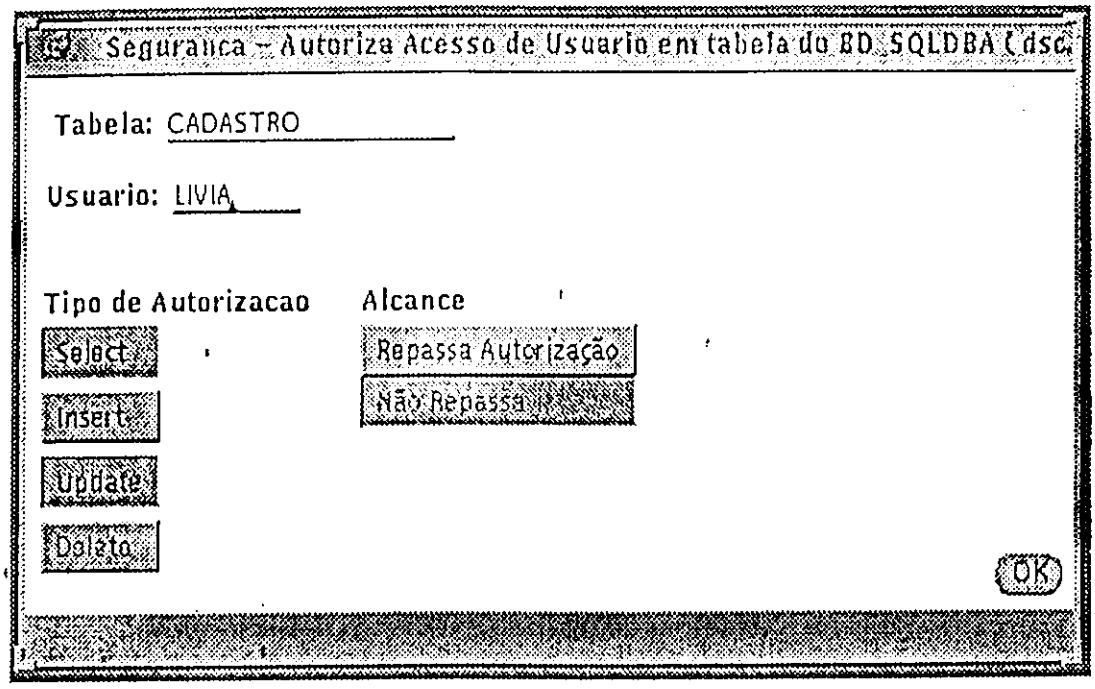

Figura A6 Autorizando acesso em tabela

A.3.1.3 *Remove acesso em tabela*

Esta operação permite que sejam removidos acessos de usuários sobre tabelas.

Ao ser digitado o nome o usuário, todas as tabelas a que ele tem acesso e, os privilégios sobre as mesmas, aparecerão em uma lista, o DBA seleciona a tabela com o mouse, e em seguida especifica qual o privilégio a ser removido no tipo de autorização, como mostra a figura A7.

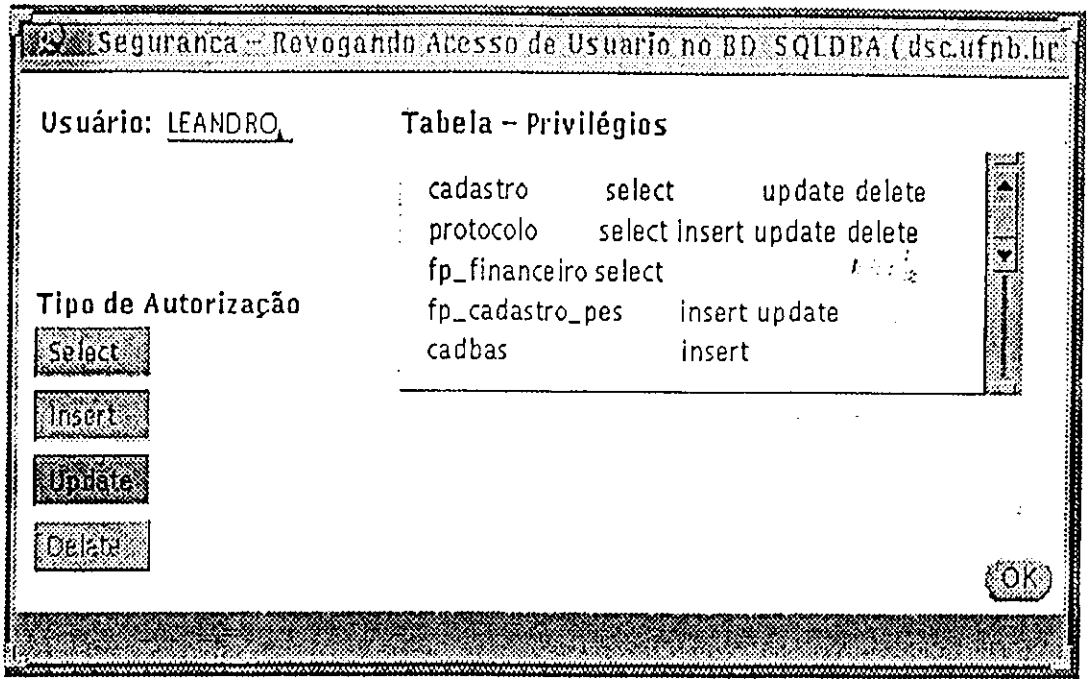

Figura A7 Operação que remove privilégios de usuários em tabelas

A.3.2 Manutenção

Uma outra responsabilidade de um administrador de banco de dados (DBA), é decidir a estrutura de armazenamento e a estratégia de acesso aos dados do banco de dados. O objetivo do comando manutenção, é fornecer operações que permitam a definição dessas estruturas, que podem ser: criar, alterar ou deletar uma tabela, e escolhendo índice, poderá fazer as operações de criar, alterar ou deletar um índice, Figura A8 e A9 e A10.

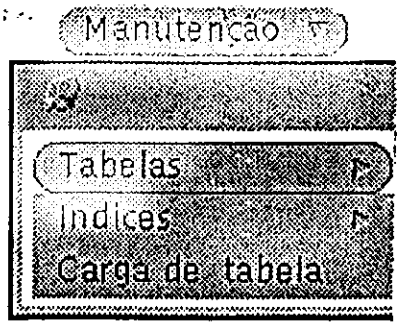

Figura A8 Operações do comando manutenção

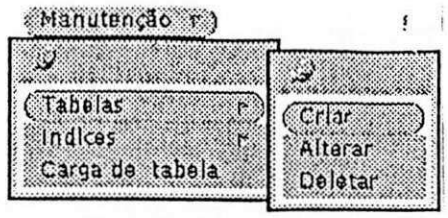

Figura A9 Comandos da operação tabela

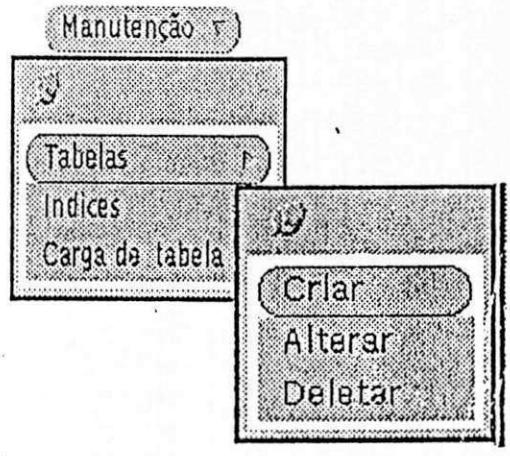

Figura A10 Comandos da operação índice

A.3.2.1 *Criar tabela*

O efeito da operação criar tabela, é a criação de uma tabela nova e vazia no BD.

O DBA deverá informar o nome da nova tabela e teclar **<enter>**, o sistema irá apresentar as áreas lógicas disponíveis, *dbspaces* - que são multiplos de 128 e as áreas físicas onde as tabelas irão armazenar seus dados - *pools.*

Após a seleção dos *dbspaces e pool*, deverá ser informado o nome da coluna, o seu tipo e se é permitido nulos, em seguida pressionando o botão insere, será apresentada a informação dos atributos da tabela em uma lista.

Para alterar as características de uma coluna, selecione a coluna na lista de atributos de tabela, digite as informações a serem alteradas e pressione o botão altera.

Para remover uma coluna, selecione a mesma e pressione o botão remove. Confirme a operação criar tabela, pressionando o botão OK, Figura A.11.

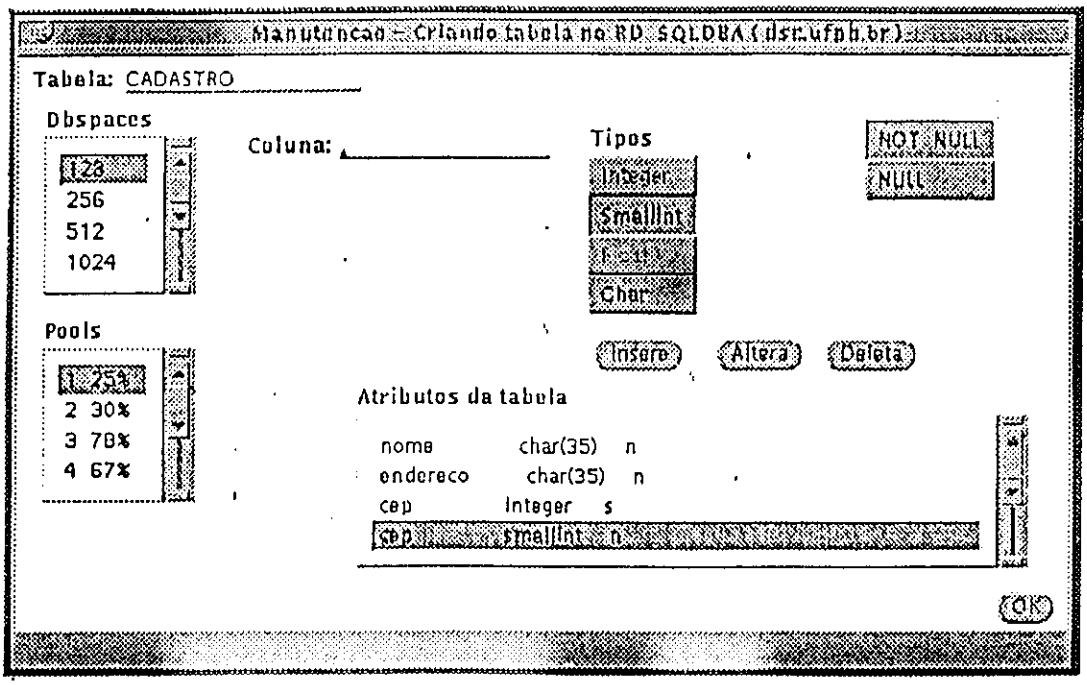

Figura A11 Cria tabela no banco de dados

A.3.2.2 *Alterar tabela*

A operação alterar tabela, oferece a mesma funcionalidade da operação criar tabela, descrita na seção A.3.2.1, a diferença é que a tabela já existe no banco de dados, e algumas colunas poderão sofrer alterações em suas caracteristicas, como mostra a figura A.12.

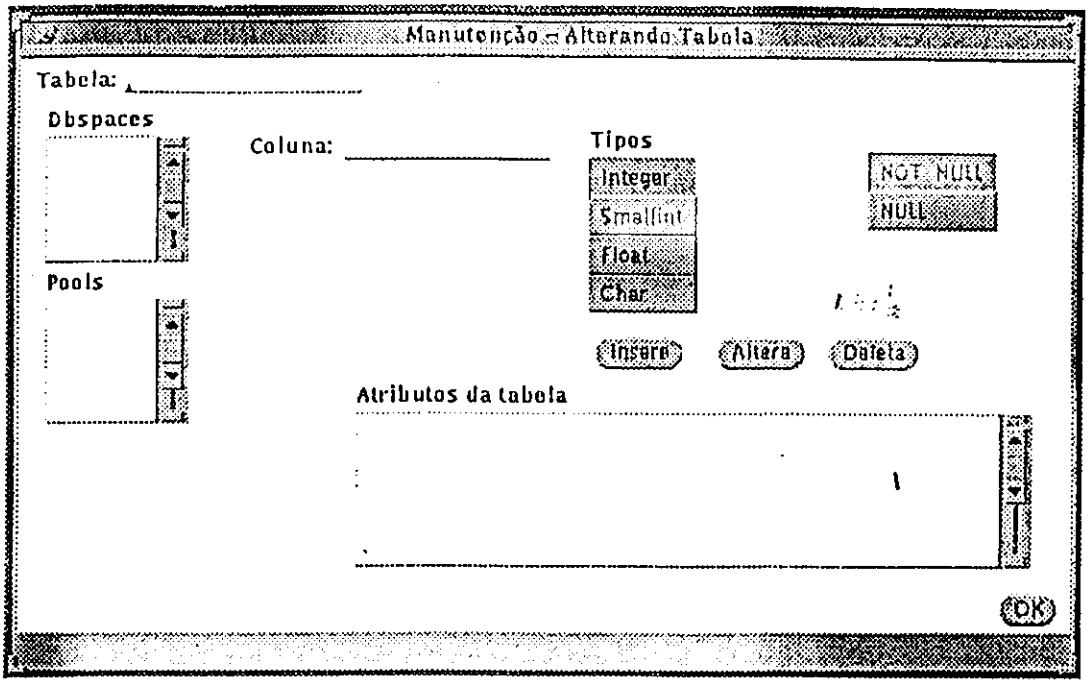

Figura A12 Altera tabela existente no BD

A.3.2.3 *Deletar tabela*

Esta operação permite que uma tabela seja removida do banco de dados. Após a digitação do nome da tabela, uma lista é apresentada contendo os atributos dessa tabela, assim, o DBA pode ter certeza se é a tabela desejada, Figura A.13.

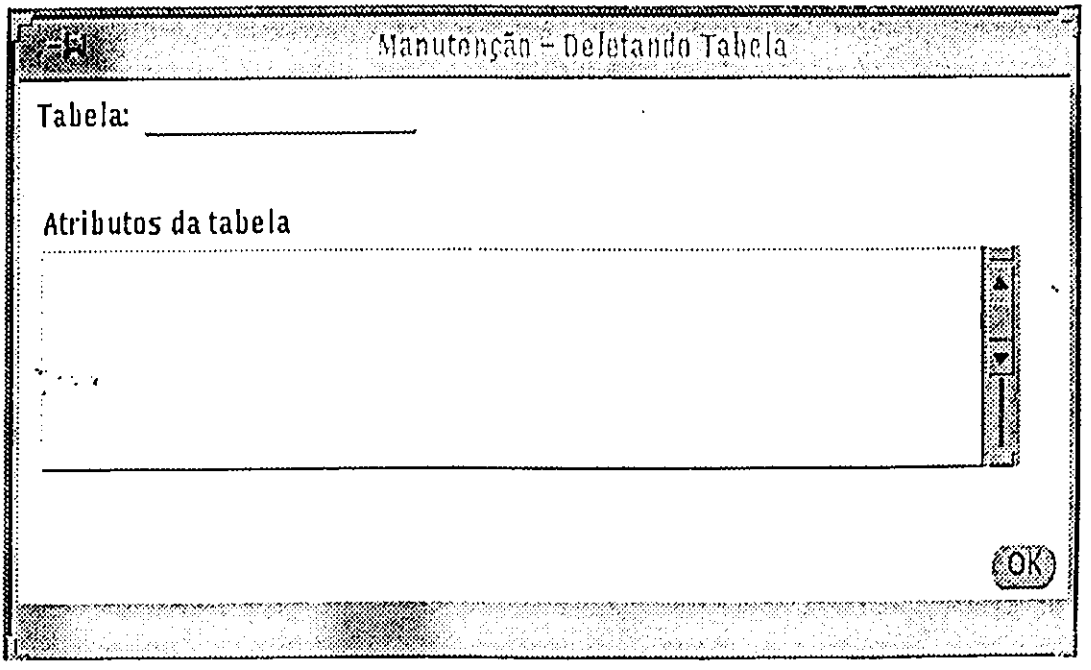

Figura A13 Remove tabela do BD

A.3.2.4 *Criar índice*

Esta opção permite que seja definido um índice para uma tabela. Quando o nome da tabela é informada, todas as colunas pertencentes a esta tabela, serão apresentadas em uma lista, o DBA deve selecionar uma ou mais colunas da lista, Figura A.14.

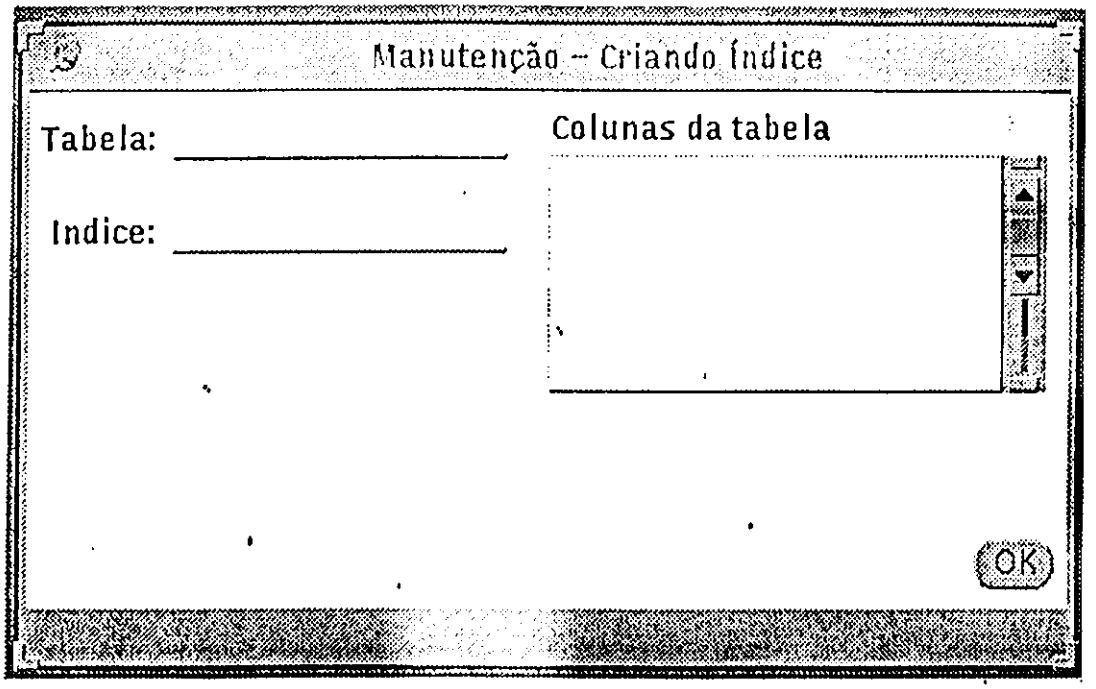

Figura A14 Criando índice para uma tabela

A.3.2.4 *Alterar índice*

Índices podem ter sua estrutura alterada através desta operação, Figura A.15. O DBA informa o nome da tabela e, todos os índices definidos sobre a mesma, serão apresentados em uma lista, para que o DBA selecione aquele que será alterado. Nesta versão do DBA-CS, todas as colunas devem ser informadas novamente.

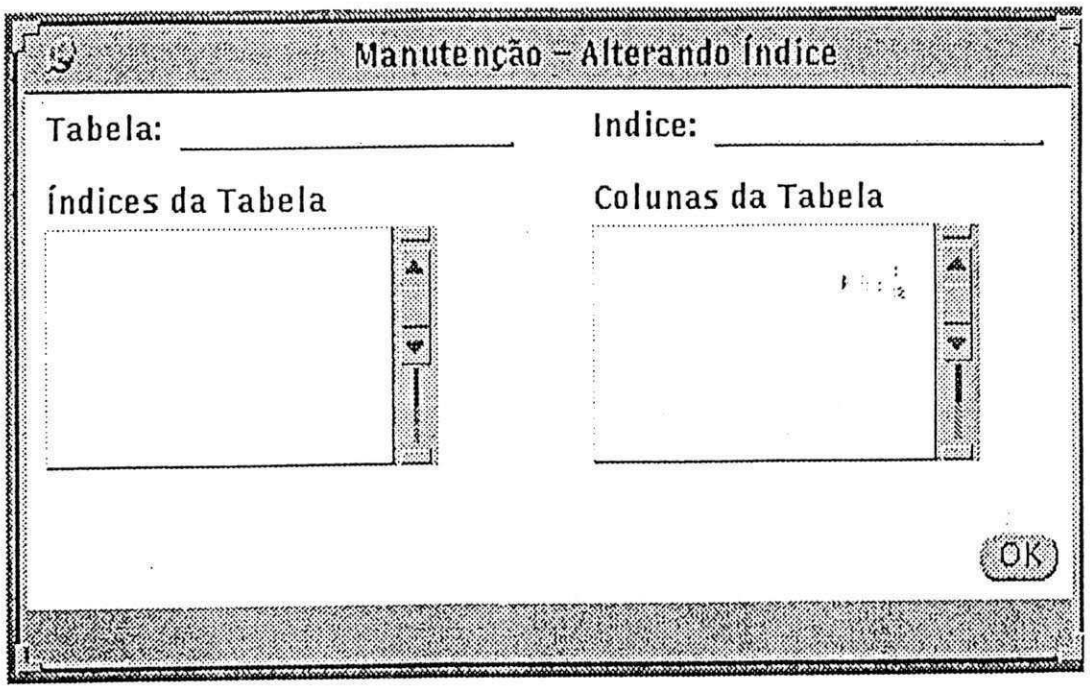

Figura A15 Altera índice de uma tabela

A.3.2.5 *Deletar índice*

Esta operação permite que uma definição de índice de uma tabela seja removida, Figura A16.

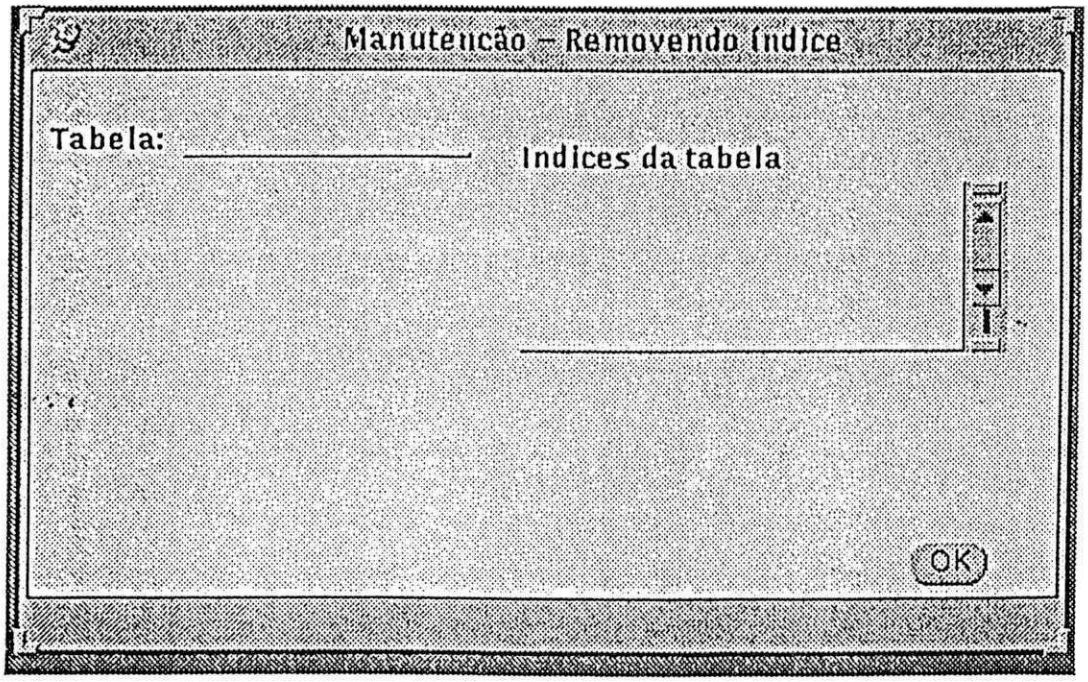

Figura A16 Remove índice de uma tabela

A.3.3 Desempenho

A função desta operação, é permitir a monitoração de desempenho do banco de dado, ou seja, analisar as possíveis causas de degradação de uma aplicação, tabela ou do SGBD, Figura A.17.

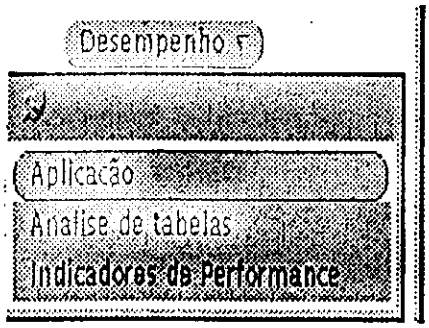

Figura A17 Operações do comando desempenho

A.3.3.1 *Aplicação*

Esta operação permite que seja feita uma análise das possiveis causas de degradação de desempenho de uma aplicação que manipula uma ou mais tabelas do banco de dados. Ao ser informardo o nome da aplicação, o DBA-CS apresenta duas listas, uma delas mostra as tabelas que esta aplicação manipula, juntamente com os índices que foram definidos para cada tabela. O *status* do índice pode assumir um dos valores: (w significa índice fragmentado, d, tabela sem índice), e a área física onde cada tabela foi definida (*pool*). A outra lista exibe o percentual de uso de cada área física, como mostra a Figura A18.

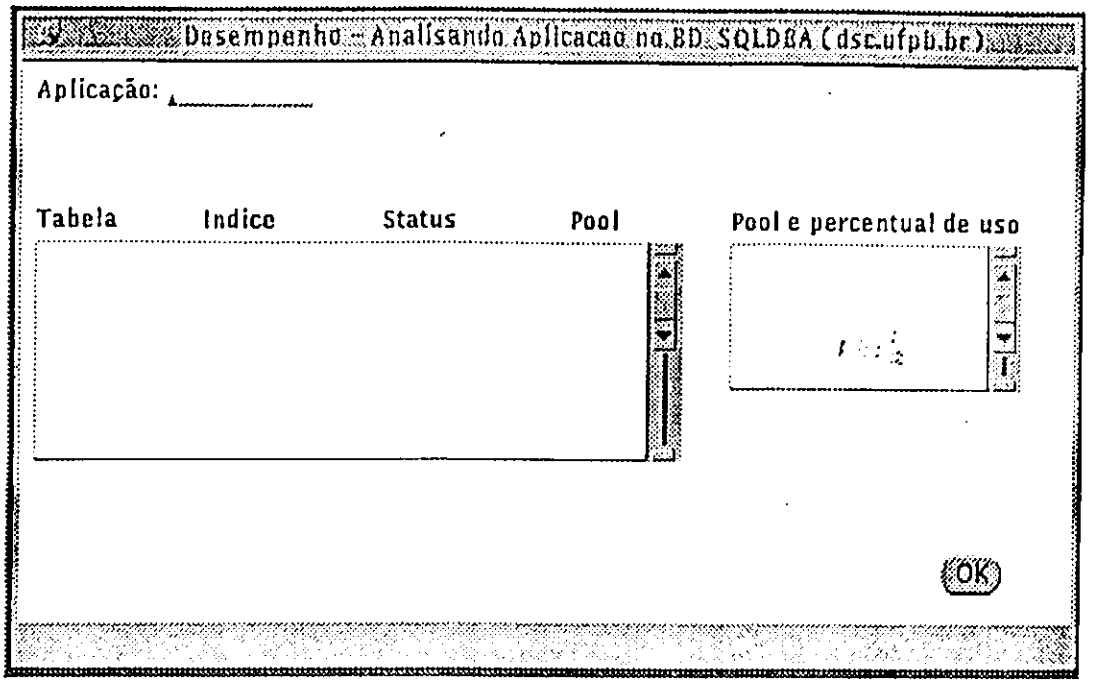

Figura A18 Analisando uma aplicação

A.3.3.2 *Análise de tabelas*

Esta operação lista todas as tabela do banco de dados que estejam com índices fragmentados. Os valores que cada índice pode assumir foram descritos na seção A.3.3.1.

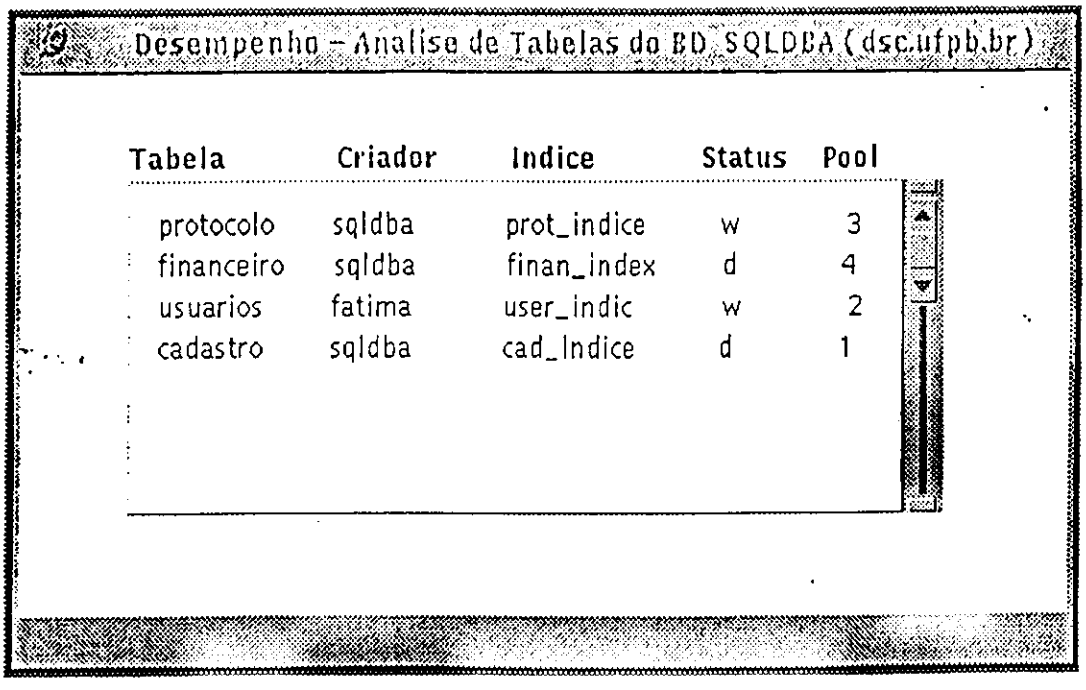

Figura A19 Analisando tabelas do BD

A.3.3.3 *Indicadores de desempenho*

Um SGBD antes de ser executado, ou seja, ficar disponível para operação, requer que alguns parâmentros sejam inicializados. Tais parâmetros são responsáveis pela adequação do ambiente, garantindo boa performance do gerenciador.

Esta função é responsável pela monitoração desses parâmetros, e quando selecionada apresenta a situação atual desses valores.

Na figura abaixo, o valor 7 para a média de páginas de buffer, significa que o número de páginas de buffers para dados, satifaz os requisistos das aplicações e consultas dos usuários.

O Log com 90% de uso, indica que o DBA deve fazer uma operação para salvar os dados contidos nesse Log, antes que alguma aplicação que faça muitas atualizações no BD comece a ser executada, pois isso faria com que houvesse uma parada do BD, para fazer a operação de salvamento automática do Log.

Pode se observar que até o presente momento não foi detectado *deadlocks*, pois seu valor é zero.

Os ajustes desses parâmetros quando necessário, são realizados com SGBD fora do ar, por isso o DBA-CS não oferece rotinas para esses ajustes.

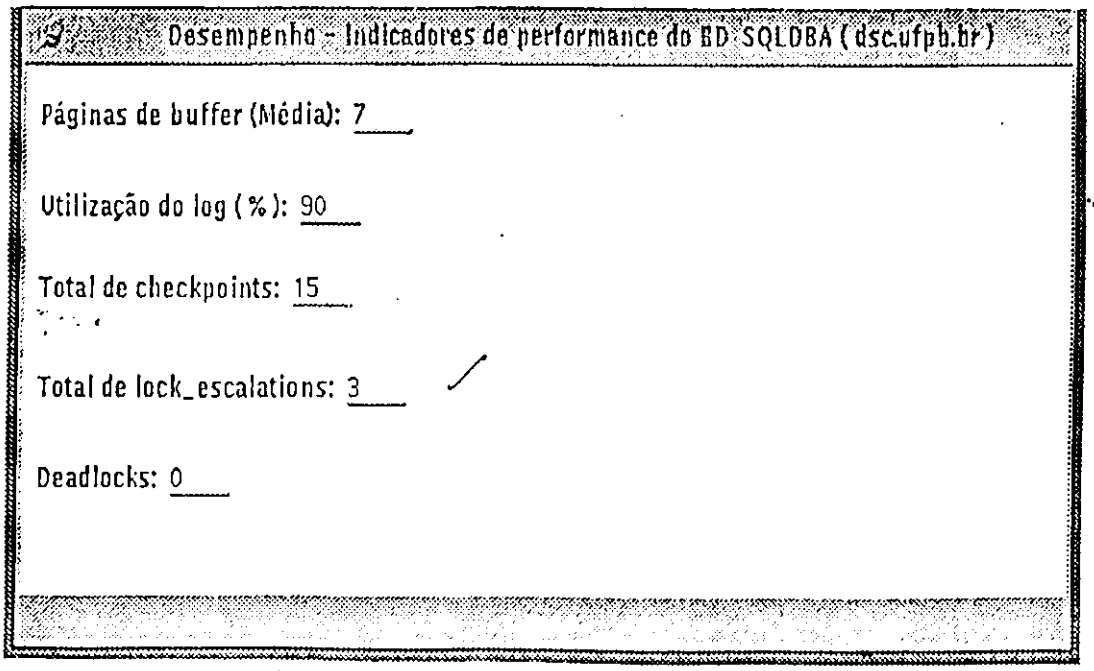

Figura A20 Indicadores de desempenho