UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

GURU - Uma ferramenta para administrar banco de dados através da Web

por

ADRIANA PAULA ZAMIN SCHERER

Trabalho de Conclusão submetido à avaliação,
como requisito parcial para a obtenção do grau de Mestre
em Informática

Profa. Dra. Nina Edelweiss
Orientadora

Porto Alegre, setembro de 2002

CIP- CATALOGAÇÃO NA PUBLICAÇÃO

Scherer, Adriana Paula Zamin
GURU – Uma ferramenta para administrar banco de dados através da Web / por Adriana Paula Zamin Scherer. – Porto Alegre: PPGC da UFRGS, 2002.
88 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, RS-BR, 2002. Orientadora: Edelweiss, Nina.

1. Administração de banco de dados. 2. Web. 3. Java 4. JDBC 5. Tomcat 6.Oracle 7. SQL Server I. Edelweiss, Nina. II. Título.

Agradecimentos

À Profa. Dra. Nina Edelweiss pela oportunidade que me ofereceu ao aceitar ser a orientadora deste projeto e desta forma contribuir para a construção de um sonho.

Ao meu marido Daniel por permanecer ao meu lado e ter me convencido de que outros caminhos eram possíveis quando o único que eu via era a desistência do sonho.

À toda minha família e em especial aos meus pais, Antônio e Laureta por terem me ensinado que o verdadeiro bem que uma pessoa possui é o conhecimento adquirido através dos estudos.

Aos amigos Roberto Pimenta, Marcel Augustus e André Bueno pelo incentivo e contribuição na elaboração deste trabalho.

Finalmente, ao Exército Brasileiro, na figura de todos os meus chefes e companheiros, que permitiram algumas ausências as quais tornaram possível o desenvolvimento deste projeto.

Sumário

Lista de Abreviaturas

API	Application Program Interface
BD	Banco de Dados
CGI	Common Gateway Interface
DB	Database
DBA	DataBase Administrator
DML	Data Manipulation Language
E/S	Entrada e Saída
EUA	Estados Unidos da América
HTML	HyperText Markup Language
HTTP	HyperText Transport Protocol
JDBC	Java DataBase Connectivity
JVM	Java Virtual Machine
JSP	JavaServer Page
ODBC	Open DataBase Connectivity
POO	Programação Orientada a Objetos
SGBD	Sistema Gerenciador de Banco de Dados
SGA	System Global Area
SQL	Structured Query Language
URL	Uniform Resource Locator
WWW	World Wide Web

Lista de Figuras

Lista de Tabelas

Resumo

Antigamente as informações que as organizações utilizavam durante a sua gestão eram suficientemente armazenadas em arquivos. A própria aplicação era responsável pela manipulação dos dados e pela função de guardá-los de maneira segura. No entanto, a sociedade evoluiu com tamanha rapidez que as organizações começaram a gerar uma quantidade cada vez maior de informação e, também, a rapidez de acesso às informações armazenadas tornou-se cada vez mais importante.

Os antigos sistemas de arquivos tornaram-se complexos sistemas de armazenamento de informações responsáveis por gerir grandes volumes de dados, chamados Sistemas Gerenciadores de Banco de Dados - SGBD's. Devido à complexidade dos bancos de dados e à necessidade de sua operação ininterrupta surge a tarefa do Administrador, cuja função é assegurar que os bancos de dados permaneçam operantes, íntegros e rápidos. Para realizar suas tarefas o Administrador precisa contar com boas ferramentas de modo a tornar as intervenções no banco de dados rápidas e seguras.

Existem no mercado, boas ferramentas para administração de banco de dados. No entanto, são todas proprietárias, possuem custo elevado e apresentam deficiências quando o DBA e o BD estão localizados logicamente em redes de dados distintas. Para tentar resolver este problema, este trabalho se propõe a desenvolver uma ferramenta de administração de banco de dados que o DBA possa utilizar para gerenciar os bancos de dados, utilizando a Web como instrumento.

Palavras-Chave: administração de banco de dados, Web, Java, JDBC, Tomcat, Oracle, SQL Server.

TITLE: “GURU – A TOOL FOR DATABASE ADMINISTRATION THROUGH WEB”

Abstract

Some years ago the information that organizations used during its administration were sufficiently stored in files and the own application was responsible for the manipulation of the data and for storing them in a safe way. However, the society developed so fast that, organizations generated more and more information, and the access speed to stored information became more and more important.

The old file systems became complex information storage systems responsible for the management of great volumes of data, called Database Manager Systems - DBMS. A Database Administrator (DBA) becomes necessary due to the complexity of database and to the necessity of uninterrupted operation. Its function is to assure the fast and secure work of databases. To perform its tasks, the DBA needs good tools to do interventions in the database in a fast and safe way.

Several tools for database administration are available in the market. However, they are expensive and show deficiencies when the DBA and the DB are logically located in different nets. To solve that problem, this work presents an administration database tool that can be used to manage the databases, using the Web as an instrument.

Keywords: databases administration, Web, Java, JDBC, Tomcat, Oracle, SQL Server.

1 Introdução

A evolução da sociedade ocorre muito rapidamente e as formas e métodos utilizados na gestão das empresas apresentam uma diferença extremamente grande, em relação aos métodos utilizados hoje em dia. A informática é a ciência atrelada à evolução da sociedade.

Para acompanhar esta rápida evolução, várias organizações têm demonstrado interesse em implantar aplicativos que controlem e otimizem suas atividades para aumentar a agilidade de seus negócios e melhorar o seu processo de tomada de decisão, trazendo a cada dia novas necessidades, ao mesmo tempo em que põem em desuso muitas tecnologias consideradas, até então, inovadoras. Como resultado deste processo, os sistemas de informação das organizações cresceram em velocidades espantosas, trazendo a cada dia exigências diferentes sobre a maneira de armazenar e recuperar a imensa quantidade de informações.

A demanda por modelos de armazenamento mais eficientes, juntamente com a complexidade dos aplicativos das organizações e a imensa quantidade de informações que eles manipulam, faz com que a complexidade dos modelos de armazenamento de informações também cresça.

Antigamente, as informações das organizações eram suficientemente armazenadas em arquivos e a própria aplicação era responsável por manipular corretamente os dados e guardá-los de maneira segura. Atualmente, para sustentar a quantidade de informações manipuladas pelas organizações, os modelos de armazenamento evoluíram e tornaram-se complexos bancos de dados. Do modesto armazenamento em arquivos até os complexos bancos de dados atuais, houve uma migração de tarefas. O banco de dados passa a assumir toda e qualquer falha com o armazenamento das informações. Para isto, novas tecnologias, específicas da área de banco de dados foram surgindo, desde o banco de dados em rede, hierárquico, relacional e, finalmente, o orientado a objetos.

Cabe ao banco de dados a responsabilidade de gerir os grandes volumes de informações das organizações, garantir a segurança destas informações contra eventuais problemas com o sistema, além de impedir tentativas de acesso não autorizadas. Se os dados forem compartilhados por diversos usuários, o sistema deve evitar a ocorrência de resultados anômalos [SIL 99].

Devido à complexidade dos atuais modelos de armazenamento, surge uma nova tarefa: a administração. As preocupações passam a ser com o volume de informações manipuladas, com a performance e com possíveis falhas que coloquem o banco de dados fora de operação.

O administrador do banco de dados (DBA) deve ser capaz de gerenciar o espaço para armazenamento dos dados no banco, conceder privilégios de acesso às informações, registrar, monitorar os usuários e coletar estatísticas [BRA 99], tudo isso no menor tempo possível de forma que o andamento normal das atividades da organização não fique prejudicada. Para realizar suas tarefas, o DBA deve ter em mãos boas ferramentas que o auxiliem em suas atividades, pois na maioria das vezes a intervenção deve ser rápida e eficaz, já que a cada minuto que o banco de dados estiver fora de operação, a organização está perdendo um pouco de sua agilidade e competitividade no mundo dos negócios.

Existem, disponíveis no mercado, muitas ferramentas que o administrador pode utilizar para facilitar e agilizar o seu trabalho. Em [SCH 2001] foi realizado um estudo sobre os principais bancos de dados e algumas de suas ferramentas de administração e todas são excelentes quando o DBA e o banco de dados a ser manutenido estão no mesmo local.

No entanto, o crescimento natural das organizações as leva a distribuírem suas funções por unidades instaladas em locais os mais variados possíveis e que, normalmente, não possuem comunicação de dados entre si por questões de custos. A solução é criar em cada unidade um banco de dados que armazene as informações necessárias à sua gestão. Embora a organização possua um banco de dados em cada unidade isolada, o responsável continua sendo somente um administrador de banco de dados (DBA) que fisicamente está locado em uma única unidade, normalmente, a matriz.

1.1 Proposta do trabalho

O principal objetivo deste trabalho é desenvolver uma ferramenta que o administrador do banco de dados possa utilizar para resolver os problemas que porventura venham a ocorrer em bancos de dados isolados e que esteja disponível para o DBA a qualquer momento, independente de configuração.

A Web, devido à sua popularidade, tornou-se o veículo perfeito para a implementação desta ferramenta, já que hoje é possível acessá-la a qualquer momento, muitas vezes através de conexões dedicadas e conectadas 24 horas.

Para o desenvolvimento da ferramenta foi utilizada a linguagem Java, por ser a principal linguagem de programação da atualidade e possuir muitas características valiosas. Dentre suas características, a mais importante é a portabilidade que facilita a criação de programas para a Internet. Para o desenvolvimento do *front-end* foi utilizado *Java Server Pages* – JSP.

Dentro da tecnologia Web, um componente importante é o servidor que abriga as páginas. Neste trabalho é utilizado o servidor Web Tomcat que pode ser utilizado sozinho ou em conjunto com outros, disponíveis no mercado e é totalmente portável entre diversas plataformas.

Por ser o SGBD mais utilizado atualmente, o Oracle foi escolhido como o principal banco de dados a ser administrado pela ferramenta. Além do Oracle, a ferramenta acessa o banco de dados SQL *Server* que também está sendo amplamente utilizado no mercado comercial. A conexão dos bancos de dados com a aplicação é feita por uma camada intermediária chamada *Java Database Connectivity* – JDBC.

Neste texto, depois de feitas as definições e estudos dos aspectos envolvendo a tecnologia Web, são apresentadas as funções práticas da ferramenta. De uma forma geral, com esta ferramenta o administrador do banco pode acessar as tabelas do dicionário de dados para identificar e corrigir os problemas que estão afetando a operabilidade do banco de dados.

1.2 Organização do texto

As próximas seções encontram-se organizadas conforme descrito a seguir. No capítulo 2 é feita uma comparação entre a ferramenta que será desenvolvida e as distribuídas pelas grandes empresas desenvolvedoras de *software* e que estão disponíveis no mercado. O capítulo 3 é utilizado para descrever o modo de funcionamento da ferramenta. O capítulo 4 é destinado ao estudo das tecnologias que serão utilizadas para o desenvolvimento da ferramenta. Dentre as tecnologias encontram-se a linguagem Java, sendo abordados os principais motivos que levaram à escolha desta linguagem para o desenvolvimento da ferramenta, bem como um estudo sobre a tecnologia JSP e sua utilização na elaboração da ferramenta. Outro aspecto abordado no capítulo 4 é a API JDBC e o relato de sua importância para a elaboração da ferramenta. Finalizando o capítulo das tecnologias, é feito um estudo sobre o servidor Web Tomcat, apresentando suas principais características e sua funcionalidade. No capítulo 5 é apresentado o banco de dados Oracle, sendo abordadas algumas características importantes e apontados os principais aspectos que necessitam da análise do administrador. No capítulo 6 é feita uma descrição detalhada do protótipo implementado utilizando as tecnologias estudadas no capítulo 4 deste trabalho. No capítulo 7 é apresentado um estudo de caso, com aplicações práticas do uso da ferramenta. Encerrando, o capítulo 8 apresenta as conclusões finais e sugestões para trabalhos futuros.

2 Ferramentas Disponíveis no Mercado

Muitos estudos vêm sendo realizados sobre uma maneira de tentar solucionar os problemas de administrar bancos de dados remotos e algumas soluções já são apresentadas por empresas e distribuídas comercialmente. No entanto, são todas ferramentas proprietárias e que demandam um certo investimento financeiro.

O fato de várias empresas comerciais iniciarem pesquisas para desenvolver ferramentas de administração de banco de dados que ultrapassem a barreira das redes de dados, indica que a importância delas não pode mais ser contestada.

A tabela 2.1 faz um comparativo entre a ferramenta desenvolvida neste trabalho e algumas soluções apresentadas por grandes empresas produtoras de *software*, visando suprir a deficiência das ferramentas tradicionais para alcançar os bancos de dados remotos.

TABELA 2.1 – Comparativo entre a ferramenta desenvolvida neste trabalho e as demais existentes no mercado

Ferramenta / **Propriedades**	**Desenvolvida neste trabalho**	**Oracle Enterprise Manager da Oracle [1]**	**Ferramentas da Hencie [2]**
Todos os módulos da ferramenta funcionam através do navegador	Sim	Não	Sim
Há possibilidade de utilização de pacotes próprios de criptografia	Sim	Não	Não
A personalização da ferramenta através da inclusão de novos módulos é possível	Sim	Não	Não
Existe um módulo para executar comandos SQL	Sim	Sim	Sim
Os módulos da ferramenta abrangem todos os aspectos dos bancos de dados	Não	Sim	Não
A ferramenta é proprietária	Não	Sim	Sim
Acesso ao BD através da Web passando por um *firewall*	Sim	Não	Desconhecido
É possível acessar bancos de dados de diferentes fornecedores	Sim	Não	Não

[1] ORA 2000, ORL 2001, ORC 2000, ORM 2000

[2] MAI 2002, MAM 2002, MAN 2002

3 Ambiente de Utilização da Ferramenta

Em sua grande maioria, as ferramentas disponíveis para administrar bancos de dados são muito eficientes quando utilizadas localmente, pois elas foram projetadas para acessar bancos de dados que estão dentro de uma mesma rede de dados. Mas elas se tornam inúteis quando a empresa possui unidades distribuídas separadas física e logicamente umas das outras.

O crescimento natural das organizações as leva a distribuírem suas funções por unidades instaladas em locais estratégicos para o negócio da empresa e que, normalmente, não possuem comunicação de dados entre si por questões de custos. Mas, as unidades também geram informações que são importantes para a sua gestão. A solução é criar em cada unidade um banco de dados que armazene suas informações. No entanto, devido ao alto grau de profissionalização e especialização necessárias à função de DBA, as organizações, apesar de possuírem vários bancos de dados, contam com o trabalho de um único profissional com esta formação. Para casos como este, as ferramentas tradicionais não podem ajudar o administrador.

Mesmo quando a organização está ligada através de redes de dados, as ferramentas tradicionais podem apresentar empecilhos ao trabalho do administrador. Suponha que o DBA esteja em uma unidade localizada fisicamente distante da unidade central, conectada logicamente, mas que não possua nenhuma ferramenta de administração de banco de dados configurada. Se durante sua permanência nesta unidade, ocorrer algum problema em qualquer das bases de dados, será necessário ao administrador instalar as ferramentas tradicionais e configurar o acesso via rede de dados. Este procedimento de instalação e configuração, certamente demandará um tempo que o administrador não pode perder. Novamente, as ferramentas tradicionais de administração de banco de dados deixam a desejar.

Ainda, suponha que a organização tenha crescido a tal ponto que todas as unidades da organização estão interligadas por redes de dados através de um *link* dedicado. É comum que cada unidade realize a proteção de sua rede interna através de softwares de segurança, conhecidos como *firewalls*. Em redes protegidas por estes softwares de segurança, o trabalho do administrador se torna mais difícil ainda, pois para entrar nas diversas redes da organização para gerenciar os bancos de dados, através das ferramentas tradicionais, é necessário que todos os gerentes de segurança das redes configurem seus *firewalls* permitindo o acesso destas ferramentas. Este detalhe pode parecer insignificante, mas em momentos de crise quando, por exemplo, o banco de dados estiver inoperante ou apresentando baixa performance, minutos perdidos em configuração são transformados em prejuízos para a organização.

Para solucionar estes problemas da administração de banco de dados, a ferramenta implementada permite o acesso a bancos de dados remotos, utilizando a Web como instrumento de comunicação, estando desta forma sempre disponível para a utilização, proporcionando agilidade e rapidez ao administrador.

O uso da Web acaba com os problemas relacionados à falta de comunicação e, também, com os problemas de acesso através dos *firewalls* pois a ferramenta com sua

interface sendo acessada através do navegador, utiliza a porta 80* que, normalmente, já está liberada.

A figura 2.1 ilustra o acesso ao banco de dados pela ferramenta.

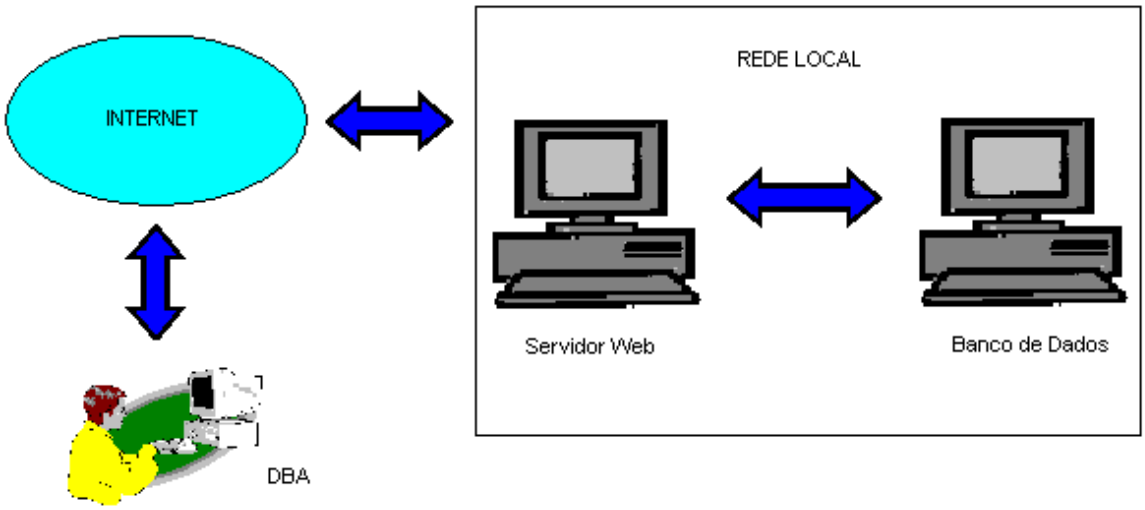

FIGURA 2.1 - Funcionamento da ferramenta para administração de BD via Web

Outro problema da administração de bancos de dados são os chamados ambientes heterogêneos, isto é, organizações que possuem bancos de dados de fornecedores diferentes. Nestes casos o administrador precisa trabalhar com várias ferramentas e em cada uma, é necessário um novo aprendizado, já que elas possuem formas diferentes de tratar os diversos enfoques dos bancos de dados e são desenvolvidas por empresas diferentes. Além do fato de que cada BD possui suas próprias estruturas que são gerenciadas de maneira individual.

Uma boa ferramenta para administrar bancos de dados necessita, além de vencer a barreira das redes internas de dados e possibilidades de acesso a bancos de dados de fornecedores diversos, auxiliar o administrador de forma específica nas tarefas de gerenciar o espaço ocupado pelos dados e avaliar o desempenho do banco de dados, permitir o fácil e rápido acesso aos principais tipos de objetos que o banco de dados possui.

No entanto, a tarefa de administrar bancos de dados engloba diversas outras atividades que em muitos casos exigem a especialidade do DBA. Para estes casos, uma boa ferramenta precisa fornecer uma área onde o administrador possa expressar seus conhecimentos em forma de comandos livres que extraem as informações necessárias para a avaliação e solução de problemas.

---

* A porta 80 é a porta padrão utilizada pelas requisições HTTP. Maiores informações, ver item 4.3.1.

4 Tecnologias Utilizadas

Este capítulo detalha as tecnologias utilizadas para o desenvolvimento da ferramenta, procurando sempre destacar as razões que motivaram a sua escolha. O item 4.1, faz uma breve descrição sobre a linguagem Java e apresenta a tecnologia JSP, utilizada para o desenvolvimento da ferramenta. O item 4.2 tem o objetivo de apresentar conceitos básicos a respeito de JDBC, sua arquitetura e a forma como esta API conecta banco de dados. O item 4.3 abordará a ferramenta Tomcat nos seguintes aspectos: principais conceitos, configuração e arquitetura.

4.1 A linguagem Java e a tecnologia JSP

Neste item são descritas a linguagem Java e a tecnologia JSP, usadas no desenvolvimento da ferramenta.

4.1.1 Java

Java é uma linguagem de programação orientada a objetos [CHA 99] que foi projetada para ser portável entre diferentes plataformas e sistemas operacionais. Está modelada conforme a linguagem de programação C++ e inclui recursos especiais que a tornam ideal para programas na Internet.

A Internet foi desenvolvida há mais de três décadas, financiada pelo Departamento de Defesa dos EUA. Originalmente projetada para conectar os principais sistemas de computadores de algumas universidades e organizações de pesquisa, hoje a Internet interliga milhões de computadores no mundo todo. A explosão da Internet ocorreu com a introdução da *World Wide Web* (WWW), que permite a usuários de computadores localizar e visualizar documentos baseados em multimídia sobre quase todos os assuntos imagináveis. Desta forma, a Internet tornou-se uma ferramenta poderosa que todas as organizações estão utilizando para tornar seus produtos acessíveis a milhares de pessoas que diariamente utilizam a Internet como meio de trabalho e passatempo.

A linguagem Java possui características [CES 96, CHA 99] que a tornaram uma das principais linguagens de programação da atualidade e que motivaram sua escolha para a programação da ferramenta. São elas:

- orientada a objetos;
- parecida com C++;
- compilada;
- independente de plataforma;
- é uma linguagem segura e robusta;
- suporta concorrência.

4.1.2 A tecnologia JSP

JavaServer Pages – JSP – é uma tecnologia [FIE 2000] baseada em Java que simplifica o processo de desenvolvimento de páginas Web dinâmicas. Com JSP, os projetistas da Web e programadores podem rapidamente incorporar elementos dinâmicos em suas páginas usando Java embutido e algumas *tags* de marcação simples. Estas *tags* fornecem ao projetista de HTML um meio de acessar dados e lógica de negócios armazenados em objetos Java sem ter que dominar as complexidades do desenvolvimento de aplicações.

Os arquivos JSP são arquivos texto, normalmente com a extensão .jsp, que substituem as páginas HTML tradicionais. Os arquivos JSP contém HTML tradicional junto com código embutido que permitem ao projetista de páginas acessar dados do código de Java rodando no servidor. Quando a página é solicitada por um usuário e processada pelo servidor de HTTP – *HyperText Transport Protocol* – a parte HTML da página é transmitida. No entanto, as partes de código da página são executadas no momento em que a solicitação é recebida e o conteúdo dinâmico gerado por este código é unido à página antes de ser enviado para o usuário. Isto propicia uma separação dos aspectos de apresentação HTML da página, da lógica de programação contida no código.

4.1.2.1 A tecnologia de conteúdo dinâmico

As solicitações mais simples da Web, as chamadas páginas estáticas, exigem do servidor Web apenas a localização do arquivo correspondente ao documento solicitado, respondendo ao navegador com o conteúdo do arquivo. A Web evoluiu e hoje, a maioria dos dados fornecidos tem uma natureza dinâmica, isto é, a informação fornecida está baseada em constante mudança, podendo ser personalizada para cada usuário em particular. A alteração de comportamento das páginas armazenadas no servidor Web exige que o servidor realize algum processamento adicional além da solicitação correspondente, para que o conteúdo dinâmico seja montado.

Os servidores de HTTP mais antigos não possuíam mecanismos para gerar páginas dinâmicas e a solução era chamar outros programas para traduzir as solicitações de conteúdo dinâmico. A primeira tecnologia para conteúdo dinâmico foi a *Common Gateway Interface* – CGI – que especificava um mecanismo para servidores da Web passarem as informações da solicitação para programas externos, os quais eram rodados pelo servidor da Web para gerar respostas no tempo de execução.

A tecnologia de CGI para geração de conteúdo dinâmico na Web apresenta um problema de sobrecarga no servidor que limita a sua aplicabilidade para uso em grande escala. Conforme já foi mencionado anteriormente, os programas de CGI rodam fora do servidor Web, o que significa que um novo processo deve ser iniciado para executar um programa de CGI. Este procedimento causa uma sobrecarga associada à criação e comunicação com o processo separado, e cada processo precisa de sua própria cota de recursos de memória da máquina local. Além do mais, os programas de CGI são projetados para tratar de apenas uma única solicitação, terminando depois que eles retornam seus resultados para o servidor Web. Portanto, para cada solicitação de conteúdo dinâmico o servidor deve iniciar um novo processo para rodar o programa CGI, enviar para ele as

informações da solicitação, esperar pelos resultados, e depois passar estes resultados de volta na sua resposta para o navegador.

Principalmente em função desta deficiência, muitas outras tecnologias para conteúdo dinâmico foram surgindo, como por exemplo, a *Active Server Pages* (ASP) da Microsoft, o ColdFusion da Allaire e a JSP que será apresentada em detalhes a seguir, por ser a tecnologia escolhida para o desenvolvimento da ferramenta.

4.1.2.2 As vantagens da tecnologia JSP

Como um sistema para geração de conteúdo dinâmico, JSP oferece muitas vantagens. Inicialmente pode ser citado o fato de JSP ser uma tecnologia baseada em Java. Herda todos os benefícios nativos de Java, inclusive ser portável a todas as plataformas que aceitam uma *Java Virtual Machine* - JVM.

A seguir serão discutidas as principais vantagens da tecnologia JSP.

Performance

Conforme já foi mencionado no item 4.1.2.1, o principal problema da tecnologia CGI é o fato de que as requisições de conteúdo dinâmico são tratadas por um novo processo que o servidor Web cria, fato este que gera uma sobrecarga no servidor.

Na tecnologia JSP, quando o servidor recebe uma requisição, ele a encaminha para um processo especial dedicado, chamado *servlet*. Há apenas um *servlet* que trata todas as requisições, ele é iniciado quando o servidor é iniciado e continua rodando até que o servidor HTTP seja desligado. As características deste processo serão apresentadas posteriormente.

O processo *servlet* tem a capacidade de tratar múltiplas requisições ao mesmo tempo através de um encadeamento Java, conforme pode ser visto na figura 4.1. Esses encadeamentos são similares aos processos, uma vez que muitos encadeamentos podem rodar simultaneamente dentro de uma JVM. Os encadeamentos causam menos sobrecarga para serem criados e destruídos do que causam os processos. Por exemplo, os processos quando são criados, herdam a cópia da memória do processo pai.

O fato de todas as solicitações JSP e *servlet* serem tratadas pela JVM facilita o compartilhamento de recursos, o que resulta em um aumento de performance.

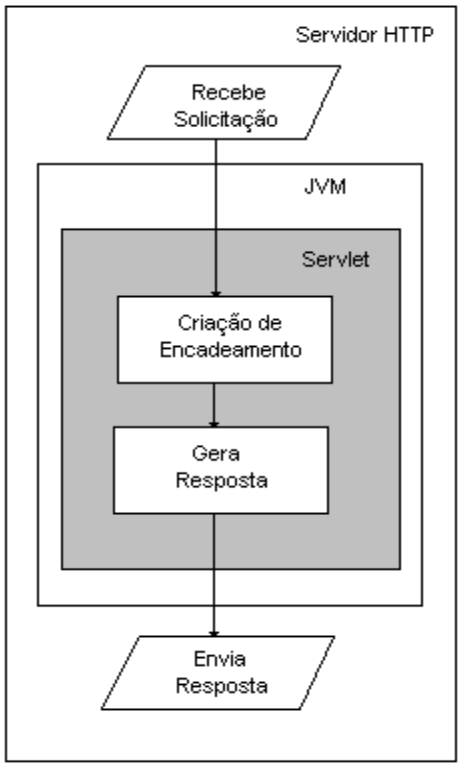

FIGURA 4.1 – Funcionamento do *Servlet*

Reutilização do código

Dentro do código JSP é possível gerar código dinâmico, ao incluir código fonte Java nas páginas Web, e manipular objetos armazenados no servidor, através de *tags* HTML. Esses objetos que são armazenados no servidor, são chamados de *JavaBeans*.

Os *JavaBeans* são objetos escritos em Java que seguem um padrão de convenções utilizadas para promover a modularidade e reusabilidade [FIE 2000] do código. Os *JavaBeans* encapsulam comportamento, funcionalidade ou dados relacionados que podem ser usados e reutilizados em múltiplos contextos sem que seja preciso conhecer os detalhes de sua programação interna.

O resultado destas características é o fato de que os *JavaBeans* podem ser conectados e combinados uns aos outros e, também, serem chamados de componentes.

A principal vantagem da programação baseada em componentes é a reusabilidade. Os componentes são módulos isolados e os programadores não precisam conhecer suas relações para poderem utilizá-los. Esta facilidade de utilização é que aumenta a sua reusabilidade.

A vantagem da reusabilidade é a produtividade. Se um componente já estiver disponível para ser utilizado, ele não precisa ser reescrito, depurado e manutenido. Além do mais, um componente não é atrelado ao contexto, por exemplo, um *JavaBean* pode ser utilizado dentro de um *servlet*, de um *applet* ou numa página JSP.

Separação da apresentação e implementação

A utilização de *JavaBeans* dentro de uma página JSP garante a possibilidade de separação entre a apresentação dos dados e a programação. A vantagem desta separação é que alterações em uma parte não implicam em alterações na outra. A forma como as informações são apresentadas pode ser alterada sem que haja necessidade de alterar o código Java. Da mesma forma, a implementação pode ser reescrita, desde que não altere a interface do componente, sem que a página JSP sofra alterações.

Para a construção de páginas JSP com separação de código, dois requisitos devem ser seguidos pelos programadores:

- primeiro requisito para a separação de código é escrever a página JSP sem código Java. O ideal é que a página JSP contenha apenas *tags* de chamadas de *JavaBeans*, deixando para estes a implementação do código Java;
- segundo requisito é não haver código HTML dentro do objeto *JavaBean*, isto é, manter o código da apresentação fora do código de implementação.

Em algumas situações há necessidade de haver interação entre a página JSP e o objeto *JavaBean*. Pode parecer necessário incluir dentro da JSP o código Java de interação. No entanto, *JavaServer Pages* possui uma forma alternativa de resolver este impasse. Podem ser desenvolvidas *tags* personalizadas que representem as interações através de HTML. A figura 4.2 ilustra a separação dos códigos.

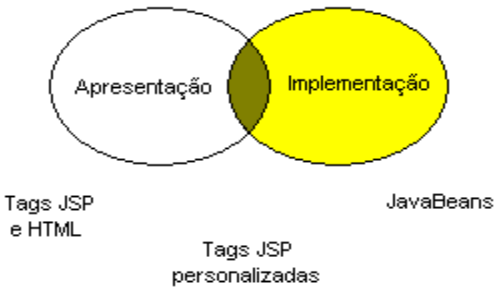

FIGURA 4.2 – Representação da separação do código de apresentação e implementação [FIE 2000]

Divisão do trabalho

O resultado da separação de códigos é a divisão de trabalho no desenvolvimento e manutenção das páginas de conteúdo dinâmico na Web.

Devido à complexidade dos atuais sistemas, é difícil para um mesmo profissional possuir conhecimentos e habilidades suficientes para ser um bom projetista e um bom programador. Desta forma, as equipes para desenvolvimento de aplicações Web são compostas por projetistas de software com grandes conhecimentos em HTML e por bons programadores que dominam a linguagem Java.

Outra vantagem da divisão do trabalho, além da especialização dos profissionais, é que os trabalhos podem ser realizados paralelamente. Enquanto a equipe de projetistas está trabalhando no *layout* das aplicações, os programadores estão desenvolvendo a lógica dos negócios.

O benefício da divisão do trabalho é explícito, não somente no desenvolvimento, mas também durante a manutenção das aplicações, já que o trabalho pode, muitas vezes envolver somente uma das especialidades, enquanto a outra continuar seu trabalho sem interrupção. Por exemplo, se houver necessidade de manutenção na interface de modo a tornar a aplicação mais amigável, somente o projetista estará envolvido, enquanto que o programador continuará com o andamento normal de suas atividades.

4.1.2.3 Como JSP funciona

O processamento de JSP inicia com uma solicitação por um documento JSP. Essas solicitações são indicadas pela URL que, normalmente, não obrigatoriamente, possui uma extensão .jsp.

Servlets

Conforme já foi visto no item 4.1.2.2, os *servlets* são processos que estão relacionados aos servidores HTTP para atender as requisições dos usuários.

Existe um conjunto de padrões de URL que estão associados para serem tratados pelo processo *servlet*. Desta forma, sempre que o servidor HTTP receber uma solicitação de URL que corresponde ao conjunto de padrões, ele encaminha esta solicitação para o *servlet* tratar. O *servlet* trata a requisição e a encaminha para uma instância de classe que fará o processamento correspondente.

Quando o servidor empacota a requisição, ele envia junto todos os dados da solicitação – URL, origem da solicitação, parâmetros e seus valores – em um objeto Java. Para retornar a resposta, um objeto Java semelhante é construído.

As classes de *servlet* são responsáveis pela definição dos métodos de serviço para tratar dos vários tipos de solicitações HTTP [FIE 2000], incluindo um método para tratar de solicitação GET de HTTP e um método para tratar de solicitações POST de HTTP. Os objetos construídos pelo *servlet* para representar uma única solicitação, e sua resposta, são passados como argumentos para estes métodos, que são então chamados pelo *servlet*, de solicitação em solicitação.

A partir de um objeto de solicitação, um método de serviço acessa as propriedades desta solicitação e realiza os processamentos necessários para construir a resposta que é encapsulada no objeto de resposta. A página HTML que engloba a resposta é gravada no fluxo de saída associado ao objeto resposta. Depois que o método de serviço tiver terminado de rodar, o *servlet* envia o conteúdo do objeto de resposta de volta para o servidor HTTP, e este envia a resposta de volta para o navegador da Web que submeteu a solicitação.

JavaServer Pages

A execução de *JavaServer Pages* começa com uma solicitação por uma página JSP. O processamento é feito nas *tags* JSP que estão na página, a fim de gerar conteúdo dinâmico. A saída deste processamento, juntamente com a HTML estática, deve ser retornada para o navegador Web.

O componente principal de uma implementação baseada em *servlets* de *JavaServer Pages* é um *servlet* especial chamado de compilador de página. Todas as solicitações com URLs que possuem a extensão JSP são atendidas por este *servlet*, que transforma o componente *servlet* em componente JSP. A tarefa do compilador de página é, para cada página JSP, compilá-la em um *servlet* específico de página, cujo propósito é gerar o conteúdo dinâmico determinado pelo documento JSP original.

Toda vez que o servidor HTTP receber uma solicitação JSP, ela é enviada para o componente JSP, que chama o *servlet* compilador de página. Na primeira vez que é feita uma solicitação para um determinado arquivo JSP, o compilador de página compila o arquivo JSP em um *servlet*.

O compilador de página vasculha o arquivo fonte em busca de *tags* e procede da seguinte maneira:

- para as *tags* JSP é gerado o código fonte Java que será executado para gerar a saída desejada;
- as *tags* HTML estáticas são gravadas sem modificação e na seqüência original no fluxo de saída;
- as *tags Beans* são traduzidas no objeto correspondente;
- os *scripts* são transferidos da forma como se apresentam originalmente.

O conteúdo dinâmico é, então, misturado com o conteúdo estático de forma a ser colocado no local correto. Este código fonte é usado para escrever os métodos de serviço para um *servlet*, de modo que ao executá-lo, a saída tenha o resultado esperado. Depois que todo o código *servlet* for construído, o compilador de página chama o compilador Java para compilar o código e adicionar o arquivo de classe no diretório correto. Quando o *servlet* de página estiver pronto, o compilador de página o chama para gerar a resposta para a solicitação. Todos estes passos – análise, geração do código e compilação – provocam uma sobrecarga no servidor. No entanto, este procedimento só é necessário na primeira vez em que uma página for solicitada. Posteriormente, todas as solicitações são passadas diretamente para o *servlet* de página. Esta afirmação é verdadeira enquanto o conteúdo da página JSP original não for alterado.

Para garantir que uma página JSP não precise ser recompilada, o primeiro passo do compilador de página é verificar a hora do arquivo compilado para determinar quando ele foi criado ou modificado. Em seguida, irá verificar a hora no *servlet* de página compilado à procura da página. Se nenhum *servlet* compilado for encontrado, ou se a hora no arquivo JSP for mais recente do que a hora no *servlet* de página compilado, então um novo *servlet* deve ser gerado. No entanto, se o *servlet* compilado for mais novo do que o arquivo JSP, significa que nenhuma alteração foi feita e o controle é transferido para o *servlet*, para que

o processamento da solicitação seja concluído. O processo de execução pode ser visto na figura 4.3.

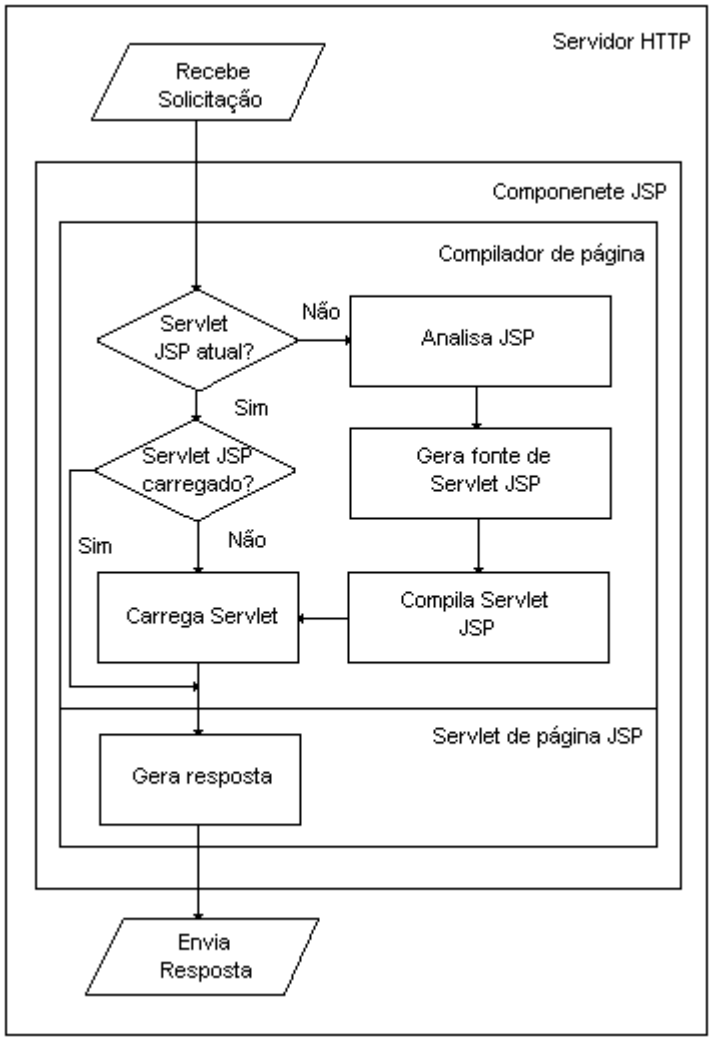

FIGURA 4.3 – Processo do servidor para criação e execução de *servlet* JSP [FIE 2000]

4.1.3 Conclusões

O principal objetivo deste estudo foi apresentar a linguagem de programação Java e a tecnologia *JavaServer Pages*.

Hoje em dia, é praticamente impossível pensar em desenvolver sistemas para Web sem logo imaginar como seria a programação Java. É uma linguagem que não está vinculada a nenhum sistema operacional ou hardware e suporta páginas dinâmicas com muita facilidade. Pode-se dizer que Java é a linguagem da Web.

Através do JSP é possível projetar um sistema que atenda aos requisitos de performance e reusabilidade de código, e que haja uma separação entre a apresentação visual e a lógica do negócio. Desta forma, possibilita a construção de *sites* eficientes e organizados.

Portanto, tanto Java quanto JSP foram as tecnologias escolhidas para a implementação da ferramenta, pois, conforme pôde ser acompanhado no desenvolver do item, a combinação das duas é, atualmente, a melhor opção para desenvolvimento de sistemas baseados em Web.

4.2 JDBC

Quando ocorreu a explosão dos *applets* baseados na linguagem Java, logo surgiu a necessidade de acessar os bancos de dados para que esta nova tecnologia fosse validada como viável para construção de sistemas, de forma que eles pudessem concorrer com a plataforma cliente-servidor utilizada até então.

Para resolver este problema, foi desenvolvida uma API padrão para conectar aplicações desenvolvidas em Java a banco de dados, chamada de *Java Database Connectivity* - JDBC. Esta API foi projetada para implementar a funcionalidade do padrão ODBC da Microsoft [INF 98], de tal forma que os drivers ODBC e códigos já existentes pudessem ser utilizados no desenvolvimento de aplicações e *drivers* JDBC.

4.2.1 O que é JDBC?

Segundo as definições elaboradas por [RAM 2001, ORE 99, HÜB 2001, FAQ 2001], JDBC é uma camada de abstração que permite a um programa Java utilizar uma interface padrão para acessar um banco de dados relacional através da linguagem SQL. A especificação do JDBC foi baseada no padrão X/Open SQL *Call Level Interface* e está de acordo com o ANSI-2 SQL.

A utilização de JDBC no desenvolvimento de sistemas, tem as seguintes vantagens [HÜB 2001]:

- a API para programação do sistema é a mesma para qualquer SGBD, não havendo necessidade de desenvolver aplicações “amarradas” a um banco de dados específico;
- permite a construção de páginas Web que acessam BD, pois dispensa a configuração da máquina cliente;
- mantém a independência de plataforma da linguagem Java, isto é, a aplicação roda em qualquer sistema operacional acessando qualquer banco de dados.

A idéia do JDBC [FAQ 2001] é que o cliente, com o seu navegador, carrega uma página HTML que contém conteúdo dinâmico e que permite a entrada de dados. A página carregada a partir do servidor Web é executada, podendo acionar a interface JDBC, o que permite a conversação com um banco de dados relacional existente no servidor, possibilitando a troca de dados entre o cliente e o servidor. Enquanto a página – *applet* ou *servlet* e *JavaServer Pages* – é responsável pelo acesso ao banco de dados, pela formatação

e apresentação dos dados, o SGBD é responsável pelo armazenamento e recuperação das informações.

Já foi comentado anteriormente que a API JDBC foi projetada para implementar a funcionalidade do ODBC da Microsoft. No entanto, de acordo com [INF 98], JDBC não é somente “ODBC para Java”. Ao invés disso, JDBC apresenta uma abordagem robusta e orientada a objetos para acesso a banco de dados relacionais, utilizando todos os benefícios da POO. Enquanto JDBC implementa muitas das funcionalidades do ODBC, ele foi implementado de uma forma consistente em contraste com a coleção de funções e estruturas de dados complexas do ODBC.

As funções e estruturas do ODBC são escritos na linguagem C, enquanto que o JDBC é um conjunto de classes Java para programação com banco de dados. Outra importante diferença [INF 98] entre as duas API’s é que o JDBC, sendo orientado a objetos, encapsula as funcionalidades de banco de dados em seu conjunto de classes. Portanto, estas classes podem instanciar objetos que são utilizados para interagir com o banco de dados.

4.2.2 Arquitetura JDBC

Para se conectar a um banco de dados, um programa Java utiliza uma API JDBC [RAM 2001] única que independe do BD ou do *driver* que estiver sendo utilizado. Os *drivers* para conexão e acesso aos principais banco de dados existentes são fornecidos pelos seus fabricantes ou por terceiros. O programador apenas precisa saber utilizar a API JDBC e a forma como o *driver* adequado se conecta ao banco de dados. A figura 4.4 ilustra a arquitetura JDBC.

Conforme pode ser visto na figura 4.4, a API JDBC nada mais é do que o pacote *java.sql* que deve ser importado nos programas Java que utilizam o JDBC para que possa haver a conexão com os banco de dados. O pacote *java.sql* contém as classes e interfaces necessárias para trabalhar com acesso a banco de dados relacionais.

Faz parte do pacote *java.sql* uma classe chamada *DriverManager*, peça central da arquitetura JDBC. Utilizada para abrir uma conexão com um banco de dados através de um *driver* JDBC registrado [FAQ 2001]. Quando uma conexão é iniciada, a classe *DriverManager* escolhe um *driver* de uma lista de *drivers* disponíveis para efetuar a conexão, dependendo da URL especificada. Uma vez estabelecida a conexão com sucesso, as chamadas para consultas e busca de resultados são feitas diretamente através do *driver* JDBC. A figura 4.5 apresenta a classe *DriverManager* na arquitetura JDBC.

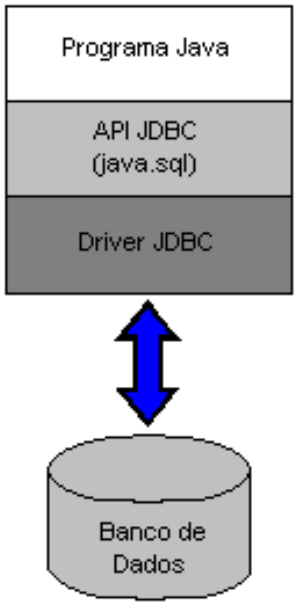

FIGURA 4.4 – Arquitetura JDBC

Um programa Java pode utilizar simultaneamente vários *drivers* JDBC, isto é, o programa pode acessar diferentes bancos de dados, bastando "carregar o *driver*". O programador não precisa saber como o *driver* foi implementado, bastando saber que ele deve ser registrado na classe *DriverManager*.

Os *drivers* JDBC se distinguem em quatro tipos e são descritos em [FAQ 2001, HÜB 2001, RAM 2001, OLI 2001]:

1. *drivers* com acesso a API parcialmente nativa – a API JDBC chama procedimentos do *driver* nativo do SGBD instalado na máquina local. Cada estação deve ter instalado o programa cliente;
2. ponte com ODBC – acesso ao banco de dados através de um *driver* ODBC convencional. A ponte JDBC aponta para a ponte ODBC;
3. *driver* com protocolo proprietário escrito em Java – a comunicação entre o cliente e o banco de dados é através do protocolo do SGBD. É o tipo mais flexível e acessa um servidor usando um protocolo neutro, por exemplo HTTP. É o servidor que estabelece a conexão com os banco de dados;
4. *driver* com protocolo nativo – o *driver* acessa diretamente o SGBD utilizando o protocolo nativo do mesmo. Por esta razão, os *drivers*, são fornecidos pelos fabricantes de banco de dados. São os chamados *thin drivers*.

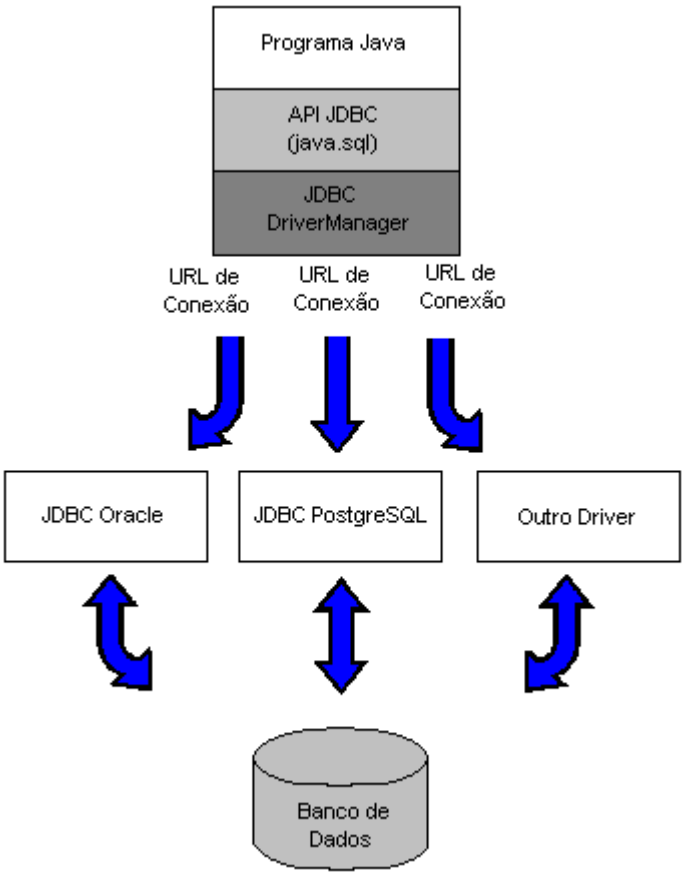

FIGURA 4.5 – A classe *DriverManager* na arquitetura JDBC

4.2.3 Conexão com o banco de dados

A conexão com o banco de dados segue sete etapas [RAM 2001], descritas a seguir:

Etapa 1 – Importar as classes JDBC

Inicialmente deve ser carregado o pacote *java.sql* para que o programa possa utilizar as classes de conexão com o banco de dados e os comandos para manipular os dados:

```
import java.sql.*;
```

Etapa 2 – Carregar o *driver* JDBC

Pode ser feito de três formas diferentes e para todas é necessário conhecer a classe que implementa o *driver*:

- carga dinâmica da classe com o método Class.forName:

```
Class.forName("sun.jdbc.odbc.jdbcOdbcDriver");
```

- instanciação de um objeto para forçar a carga da classe:

```
new sun.jdbc.odbc.jdbcOdbcDriver();
```

- atribuição a variáveis do sistema, no momento de invocação do programa Java, com a opção –D:

```
java -Djdbc.drivers=sun.jdbc.odbc.jdbcOdbcDriver program
```

Os exemplos acima estão baseados numa conexão do tipo ponte ODBC. Para utilizar outros *drivers* JDBC basta substituir a classe `sun.jdbc.odbc.jdbcOdbcDriver` pela classe que implementa o *driver* desejado.

Após a carga do *driver* é necessário registrá-lo na classe *Driver Manager* :

```
Driver drv = new sun.jdbc.odbc.jdbcOdbcDriver();
DriverManager.registerDriver(drv);
```

Etapa 3 – Especificar um banco de dados

Nesta etapa é definida a URL de conexão específica para cada banco de dados. Normalmente a documentação da URL de conexão é fornecida junto com o *driver*. Para obter a conexão, esta deve ser passada como uma *string*.

O formato de URL para conexão a um banco de dados Oracle, por exemplo, é jdbc:oracle:<*driver*>:@<servidor>:<porta>:<instância>.

Etapa 4 – Abrir a conexão

A conexão deve ser aberta através do método estático *getConnection* da classe *DriverManager*, onde o parâmetro do método é a URL de conexão e o retorno é um objeto que implementa o objeto *Connection*:

```
Connection con = DriverManager.getConnection (url, "usuário", "senha");
```

O *DriverManager* analisa a URL de conexão [PED 2001] e tenta conectar com o primeiro *driver* carregado e, se não conseguir tenta o *driver* seguinte. Caso ele não consiga estabelecer conexão com nenhum dos *drivers* carregados, então uma exceção é gerada. Uma vez inicializada a conexão, através do objeto *Connection* é criada uma linha direta com a base de dados [OLI 2001].

Etapa 5 – Criar um *Statement*

Um objeto *Statement* é utilizado para enviar comandos para o banco de dados. Os *Statements* podem ser de três tipos:

- *Statement* - utilizado para enviar comandos SQL simples, isto é, que não envolvem parâmetros:

```
Statement comando = con.createStatement();
```

- *PreparedStatement* - o comando SQL é pré-compilado e utilizado posteriormente:

```
PrepareStatement comando = con.prepareStatement(stringSQL);
```

- *CallableStatement* - utilizado para chamar procedimentos SQL armazenados no banco de dados, as *Stored Procedures*:

```
CallableStatement comando = con.prepareCall(stringSQL);
```

Nos exemplos acima, foi utilizado o objeto `con` criado na etapa 4.

Etapa 6 – Submeter um comando SQL

É importante distinguir o tipo de sentença SQL que se deseja utilizar [OLI 2001], visto que o método de enviar consultas (*query*) difere do envio de atualizações (*update, insert, delete*) na base de dados. A principal diferença é que o método da consulta retorna uma instância da classe *ResultSet* enquanto que a atualização retorna um número inteiro.

As consultas são executadas pelo método *executeQuery* do objeto *Statement*, e as atualizações pelo método *executeUpdate* do mesmo objeto. Os exemplos a seguir tornam o texto mais elucidativo:

```
ResultSet rs = comando.executeQuery("select * from EMP");
int lin = comando.executeUpdate("delete * from EMP");
```

Etapa 7 – Receber os resultados

Conforme já foi visto na etapa 6, um comando SQL pode receber como resultado um conjunto de linhas de uma tabela (chamado *ResultSet*), um número inteiro, parâmetros de saída de uma *Stored Procedure* ou, ainda, uma combinação de todos os anteriores:

- *ResultSet* - é um conjunto de registros que retornam de uma consulta. Os registros podem ser acessados seqüencialmente ou aleatoriamente. As colunas de cada registro podem ser acessadas através de um índice ou pelo nome da coluna:

```
ResultSet rs = comando.executeQuery("select * from EMP");
// next() posiciona na primeira linha, se existir
// ou posiciona na próxima linha, se existir
while (rs.next()) {
     // getString(1) : primeira coluna da tabela
     System.out.print(rs.getString(1));
     // getString(2) : segunda coluna da tabela
     System.out.print(rs.getString(2));
     // getString(3) : terceira coluna da tabela
     System.out.print(rs.getString(3));
}
```

- número inteiro - os comandos de atualização no banco de dados (*insert*, *update* ou *delete*) atingem um número de linhas na tabela. O retorno do método *executeUpdate* fornece esse número:

```
int lin = comando.executeUpdate("delete * from EMP");
```

- parâmetros de saída de *Stored Procedures* - através dos métodos da classe *CallableStatement* é possível indicar quais os parâmetros de saída, quais os de entrada e qual o tipo da *procedure*.

4.2.4 Conclusões

O desenvolvimento de produtos relacionados a Java já está ocupando boa parte do mercado de sistemas de informação e, em pouco tempo, deve chegar aos índices de linguagens tradicionais como Delphi e Visual Basic.

Java deixou de ser uma ferramenta utilizada apenas para construir páginas bonitas na Web. Está se estabelecendo como uma linguagem de programação séria, à altura das aplicações empresariais mais sofisticadas. Este fenômeno somente se tornou possível com a possibilidade destas aplicações acessarem os bancos de dados através da API JDBC.

No item 4.2 foi apresentada a API JDBC que é utilizada para fazer uma conexão entre os programas Java e um banco de dados. Esta conexão é feita de forma simples, e com poucos métodos é possível executar todos os comandos SQL ANSI 92 passando-os como parâmetros para o banco de dados.

A utilização desta tecnologia no desenvolvimento de programas Java que tem a pretensão de acessar banco de dados é o caminho natural da programação.

4.3 Tomcat

Uma peça importante da tecnologia Web é a que torna as páginas disponíveis para que as pessoas possam acessá-las através de seus navegadores – estas peças são chamadas de servidores Web.

Um servidor Web [STO 97] é um programa que roda em um computador conectado à Internet ou à Intranet. O servidor Web espera que os navegadores peçam documentos HTML. Quando recebe um pedido, ele encontra o documento HTML e o envia para o navegador que o solicitou.

Um servidor Web tem quatro funções básicas [STO 97]:

- servir páginas Web;
- rodar programas de interconexão (*gateway*) e retornar seus resultados;
- controlar o acesso ao servidor;
- monitorar e registrar estatísticas de acesso ao servidor.

O modo como essas funções são implementadas pode variar sensivelmente de um servidor para outro.

Há, ainda, uma outra característica relativa aos servidores Web que tem se tornado exigência com a evolução das tecnologias para desenvolvimento de sistemas para a Web – o suporte a *servlets* e às páginas JSP. Para fazer uso destas características avançadas que se desenvolveram sobre a plataforma Java 2, é necessário que o servidor Web suporte tais funções. Normalmente, um "motor" é incorporado ao servidor Web para que ele reconheça estas características. Dentre os diferentes motores que existem no mercado para extender as características adicionais do Java 2 provavelmente um dos projetos mais interessantes [WML 2001] liderado pela Apache Software Foundation é o projeto Jakarta no qual foi desenvolvida a aplicação Tomcat, um servidor de aplicações com as características de servir como motor de *servlets* e JSP e com a vantagem de ser gratuito.

4.3.1 Principais conceitos

Em junho de 1999 [FIE 2000, INT 2001], a Sun Microsystems anunciou que estaria entregando para a Apache Software Foundation o código fonte interno das implementações de referências das APIs de JSP e *servlet*, para uso no desenvolvimento de uma versão *Open Source* destas tecnologias. A Apache criou o projeto Jakarta com o objetivo de produzir implementações de nível mundial das especificações de JSP e *servlet*, ao mesmo tempo em que objetivou contar com a contribuição de qualquer pessoa interessada em participar do desenvolvimento e aperfeiçoamento do software, pelo fato de ser uma plataforma aberta.

O lançamento do projeto Jakarta ocorreu em dezembro de 1999 com a apresentação da primeira versão – 3.0 – do Tomcat. O Tomcat é um *servlet* e *container* JSP que implementa as especificações básicas de *servlets* e JSP desenvolvidas pela Sun. Ele inclui o servidor HTTP baseado em Java e, portanto, pode ser executado como um software independente. Mas ele possui, também, um módulo conector que suporta a integração com outros servidores Web como, por exemplo, o Apache e o Internet Information Server da Microsoft. Neste trabalho será apresentado o Tomcat como servidor independente. Os detalhes referentes ao Tomcat conectado a outro servidor não farão parte do escopo deste estudo.

O software Tomcat foi totalmente desenvolvido em Java, logo, ele possui todas as características inerentes a um programa Java, sendo a principal delas a independência de plataforma.

Como todo servidor Web, o Tomcat também precisa saber em qual das portas ele deve procurar as informações para funcionar de maneira adequada. O Tomcat, rodando como servidor independente, utiliza a porta 8080 como padrão. Para testar o seu funcionamento, basta apontar o navegador para o endereço http://server:8080/, onde "*server*" é o nome do servidor que abriga o Tomcat. Ao chamar este endereço, o navegador irá mostrar a página inicial da instalação do Tomcat que contém *links* para a documentação do software.

A porta padrão dos servidores HTTP é a 80, sendo que o Tomcat também pode responder por requisições nesta porta. Para isto basta alterar o arquivo de configurações *server.xml* no elemento *Connector*, onde está indicada a porta padrão 8080, usada para "ouvir" as solicitações HTTP, e substituí-la pela 80. A partir desta alteração e após a reinicialização do programa, o Tomcat passará a usar a porta 80 como padrão. Uma descrição mais completa do arquivo de configurações *server.xml* será feita posteriormente.

4.3.2 Configuração

O funcionamento do servidor Tomcat está totalmente baseado na configuração [APC 2001] de dois arquivos:

- *server.xml* – arquivo de configuração global do Tomcat;
- *web.xml* – arquivo de configuração dos vários *servlets*.

4.3.2.1 *server.xml*

É neste arquivo que são definidas várias configurações que controlam a operação do servidor, como, por exemplo, a porta de rede na qual o servidor ouve as respostas [FIE 2000]. O arquivo *server.xml* também identifica as aplicações da Web que deverão ser conhecidas pelo servidor.

Os vários componentes do arquivo *server.xml* estão descritos na tabela 4.1.

O componente *Context* merece mais algumas explicações, por ser ele que determina o reconhecimento das aplicações Web pelo servidor Tomcat. São parâmetros deste componente, os itens [APH 2001, FIE 2000, APE 2001] relacionados a seguir, e que podem ser definidos para cada contexto:

- *path* – especifica o prefixo de diretório para URLs que devem ser tratados pela aplicação da Web. Por exemplo, uma aplicação onde o *path* é "/guru", qualquer requisição que começar com "/guru", será processado por esta aplicação;
- *docBase* – indica onde os arquivos para a aplicação realmente estão armazenados, isto é, em qual diretório especificamente;
- *debug* – especifica o nível de detalhamento para gerar *logs* para este contexto. O padrão é o valor "0" e indica o menor nível de detalhes a serem "logados";
- *reloadable* - este parâmetro definido como verdadeiro significa que o Tomcat irá identificar alterações nas classes do contexto e automaticamente finalizará a aplicação para carregar as alterações. Esta propriedade definida como verdadeira é útil para ambientes de desenvolvimento, no entanto em ambientes de produção é prudente defini-la como falsa para evitar acidentes.

O arquivo de configuração está localizado dentro do TOMCAT_HOME/conf, onde TOMCAT_HOME é o diretório base de instalação do servidor Tomcat.

Um exemplo de arquivo de configuração *server.xml* é apresentado no anexo 1.

TABELA 4.1 – Componentes do arquivo *server.xml*

Elemento	Descrição
Server	É o primeiro elemento do arquivo *server.xml* e representa o servidor Tomcat.
Logger	O elemento *Logger* irá definir os vários objetos que geram os *logs* do servidor. Como *default* já são definidos os *loggers* para os *servlets*, para os arquivos JSP e para o próprio Tomcat.
ContextManager	Através deste componente é possível: • definir o nível de depuração usado para "logar" as mensagens; • definir os diretórios usados para a inicialização do Tomcat; • definir o nome do diretório de trabalho.
ContextInterceptor e *RequestInterceptor*	Estes componentes ficam "escutando" eventos que ocorrem no *ContextManager*. O *ContextInterceptor* escuta os eventos de inicialização e finalização do servidor Tomcat. O *RequestInterceptor* "acompanha" algumas etapas das requisições dos usuários, como, por exemplo, geração de logs.
Connector	O *connector* é o responsável pelas conexões dos usuários. Ele gerencia as requisições e respostas para os clientes. Conforme citado no item 4.3.1, este componente define a porta onde as requisições serão atendidas.
Context	Cada contexto representa a hierarquia onde as aplicações Web estão armazenadas.

Fonte: [APC 2001].

4.3.2.2 *web.xml*

De acordo com [APE 2001], neste arquivo estão descritos os *servlets* e outros componentes que fazem parte das aplicações, junto com parâmetros de inicialização e dispositivos de segurança.

O arquivo *web.xml* é também chamado de arquivo descritor de acionamento – Web *Application Descriptor*, que especifica como o conteúdo é usado para fornecer a

funcionalidade da aplicação correspondente. Por exemplo [FIE 2000], o arquivo descritor relaciona os *servlets* contidos em uma aplicação. Para cada contexto definido no arquivo *server.xml*, há um arquivo *web.xml*.

Um exemplo de arquivo de configuração *web.xml* é apresentado no anexo 2.

4.3.3 Arquitetura

Ao ser instalado o servidor Tomcat, uma árvore de diretórios é criada no computador que abriga o software. Esta árvore de diretórios é que forma a arquitetura do servidor, onde cada diretório possui uma série de arquivos que ao todo formam a estrutura do servidor Web Tomcat.

Na tabela 4.2 estão relacionados os diretórios que compõem a arquitetura do Tomcat e uma rápida descrição de seu conteúdo.

TABELA 4.2 – Arquitetura de diretórios do Tomcat

Diretório	Descrição
Bin	Contém os scripts de manipulação do Tomcat como, por exemplo, de inicialização e finalização do servidor.
Conf	Contém os arquivos de configuração, entre eles, o *server.xml* e o padrão de *web.xml* .
Doc	Contém uma miscelânea de arquivos, entre eles um *readme* e um *faq*.
Lib	Contém os arquivos que garantem o funcionamento do servidor.
Logs	Armazena os *logs* gerados durante a operação do servidor Tomcat.
Src	Contém interfaces vazias e classes abstratas que podem ser implementadas por qualquer *container* de *servlet*.
Webapps	Contém um exemplo de aplicação.
Work	Utilizado durante a operação do sistema para armazenar arquivos intermediários como os arquivos JSP compilados. Se este diretório for apagado com o Tomcat em execução ele, deixará de funcionar corretamente.
Classes	Opcionalmente criado pelo administrador do servidor para armazenar as classes das aplicações.

Todos os diretórios listados acima estão localizados sob um diretório principal chamado de TOMCAT_HOME.

Na estrutura vista na tabela 4.2 foram listados os diretórios que fazem parte da estrutura do software Tomcat, no entanto, a localização das aplicações Web não foi mencionada. Um local sugerido para armazenar as aplicações é o diretório *webapps* que já possui uma aplicação exemplo que faz parte do software Tomcat. Esta sugestão é feita por questões de organização, mas o diretório que armazena a aplicação pode ter qualquer nome e estar em qualquer lugar no disco.

Supondo o diretório *webapps* como aquele que contém todas as aplicações Web que o servidor Tomcat gerencia, dentro desta estrutura deve ser criada uma subestrutura que mantém organizada a aplicação.

A subestrutura é composta obrigatoriamente pelo diretório *web-inf* [APH 2001, GOO 2001] e dentro dele estão localizados o arquivo descritor de acionamento do contexto *web.xml*, as classes e bibliotecas da aplicação.

A figura 4.6 ilustra uma sugestão de hierarquia de aplicações gerenciadas pelo servidor Web Tomcat.

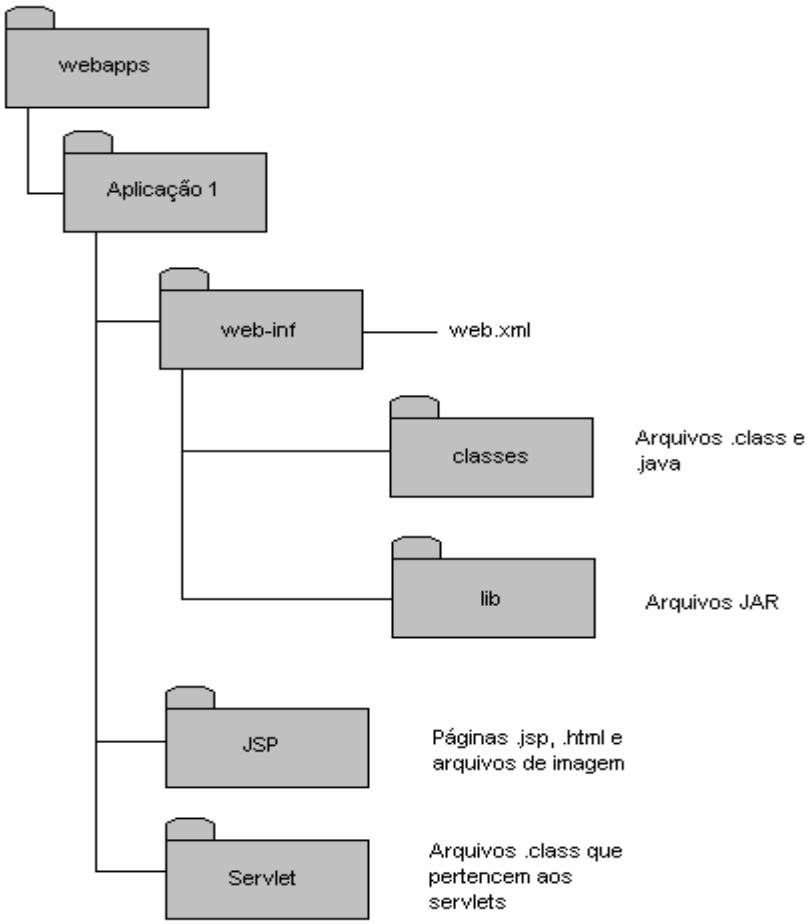

FIGURA 4.6 – Arquitetura de aplicações no Tomcat

4.3.4 Conclusões

O item 4.3 apresentou os aspectos principais do servidor de aplicações Web Tomcat. Inicialmente foram discutidos os conceitos iniciais, abrangendo um pouco da história do surgimento do Tomcat. Posteriormente foi realizado um estudo sobre sua configuração e, finalmente, apresentada a arquitetura do servidor.

Através deste estudo foi possível concluir que o Tomcat é um “motor” para *servlets* e páginas JSP que pode ser conectado a um servidor Web mais completo, como, por exemplo, o Apache, que também pode perfeitamente ser utilizado como um servidor autônomo. Esta última opção é bastante interessante, uma vez que o Tomcat é extremamente fácil de ser instalado e configurado. Além disso, ele tem a vantagem de ser gratuito e estar disponível para várias plataformas.

5 Banco de Dados Oracle

No capítulo anterior foram apresentados vários aspectos da tecnologia Web para desenvolvimento de sistemas. No entanto, de nada adiantam tecnologias fantásticas e modernas de interface, se por trás deles não houver uma ferramenta poderosa e confiável para realizar o armazenamento e gerenciamento das informações, que são o bem mais precioso de qualquer organização.

Um servidor de BD é a chave para resolver os problemas de gerenciamento de informação. Em geral, um servidor deve gerenciar de forma confiável uma grande quantidade de dados em um ambiente multiusuário de forma que vários usuários possam acessar concorrentemente os mesmos dados. Além disso, eles devem realizar estas tarefas com o máximo de performance. Um BD deve também prevenir acessos não autorizados e fornecer soluções eficientes para recuperação em caso de falhas. o BD Oracle fornece soluções eficientes para estes problemas.

Este capítulo está destinado à apresentação do banco de dados Oracle. A escolha do Oracle foi motivada pela ampla utilização que este banco de dados tem no mercado atualmente.

Segundo [SAR 2000], um servidor Oracle é um sistema de gerenciamento de banco de dados relacional de objeto que fornece uma abordagem aberta, abrangente e integrada para o gerenciamento das informações.

Um administrador de banco de dados Oracle é a pessoa responsável pela manutenção do servidor Oracle de forma que este servidor permaneça em condições de responder às requisições dos usuários. Para isto é necessário conhecer a estrutura do banco de dados, pois em casos de necessidade de qualquer intervenção o DBA deve ter consciência do que fazer.

A seguir são analisados os principais aspectos relativos ao banco de dados Oracle. Como o objetivo básico deste trabalho é implementar uma ferramenta para administração de um banco de dados Oracle, os aspectos apresentados são os que deverão ser contemplados pela ferramenta.

Inicialmente, são apresentados aspectos gerais relativos a um banco de dados Oracle cujo objetivo é promover a familiarização com termos referentes à tecnologia Oracle e, posteriormente, é apresentado o dicionário de dados do BD Oracle, que é a base para as consultas da ferramenta desenvolvida.

5.1 Estrutura do banco de dados Oracle

Segundo [ORO 2000], um BD Oracle é composto de estrutura física e lógica que são independentes uma da outra. Por este motivo, a estrutura física pode ser alterada sem alterar o acesso às estruturas lógicas.

5.1.1 Estrutura Física

A parte física [ORR 99] de um banco de dados Oracle é composta pelos arquivos gravados fisicamente no sistema operacional e que podem ser de três tipos: *datafiles*, arquivos de *redo log* e arquivos de controle.

Um arquivo do tipo *datafile* é aquele que contém as tabelas e os índices que compõem os sistemas de informação. Um banco de dados Oracle pode ser composto por um ou mais arquivos *datafile*, mas um arquivo *datafile* pode estar associado somente a um banco de dados Oracle.

As informações que estão armazenadas nos *datafiles* são recuperadas conforme a necessidade, isto é, quando surgir uma solicitação de um cliente por determinada informação que está armazenada. Ao ser recuperada a informação que está num determinado *datafile*, o Oracle a armazena em um endereço de memória alocada para o Oracle na memória real do servidor, tornando-a disponível para que os demais usuários possam acessá-la de forma mais rápida. O mapeamento da memória ocupada pelo Oracle será discutido posteriormente.

Os arquivos de *redo log* são utilizados para recuperação de um estado seguro do banco de dados em caso de falha do sistema. Cada banco de dados Oracle possui, no mínimo, dois ou mais arquivos de *redo*.

A principal função dos arquivos de *redo* é armazenar todas as alterações realizadas sobre as informações. Cada banco de dados Oracle deve ter pelo menos dois arquivos de *redo* que podem possuir cópias para serem distribuídas por diferentes discos a fim de aumentar a segurança do banco de dados.

Cada banco de dados Oracle possui um arquivo de controle. Um arquivo de controle contém ponteiros que especificam a estrutura física do BD. Dentre outras informações importantes, o arquivo de controle possui as seguintes informações [ORO 2000]:

- nome do banco de dados;
- os nomes e localizações dos *datafiles* e *redo log*;
- arquivo de *redo log* corrente;
- *time stamp* de criação do banco.

Da mesma forma que os arquivos de *redo log*, os arquivos de controle devem possuir cópias distribuídas em discos diferentes para garantir maior segurança.

Cada vez que o banco é inicializado, o arquivo de controle é lido para localizar os *datafiles* e os arquivos de *redo log*. Durante a operação do sistema, o arquivo de controle é alterado continuamente com várias informações como, por exemplo, o arquivo de *redo log* corrente. Toda e qualquer alteração física no banco de dados também é registrada no arquivo de controle, por exemplo: a inclusão de um novo *datafile* ou a criação de outro arquivo de *redo log*.

O arquivo de controle é, também utilizado nos processos de recuperação do banco de dados como um guia para remontar a sua estrutura.

Além dos *datafiles*, *redo log* e arquivos de controle, também fazem parte da estrutura física de um banco de dados Oracle o arquivo de inicialização, o arquivo de senhas e os arquivos de *log* "arquivados", cujo gerenciamento não será abordado pela ferramenta.

5.1.2 Estrutura Lógica

A estrutura lógica do Oracle é composta por *tablespaces* e esquemas. Os esquemas são coleções de objetos que podem ser tabelas, visões, seqüências e índices, entre outros. Os *tablespaces* são estruturas lógicas que agrupam os objetos que compõem os esquemas.

Pode-se dizer, ainda, que os *tablespaces* são unidades lógicas de armazenamento e são utilizados para organizar o banco de dados. Por exemplo, um banco de dados que armazena informações de um sistema de faturamento e um de pessoal, podem ser criados dois *tablespaces*: faturamento e pessoal. Todos os objetos que forem criados para cada sistema serão criados dentro do *tablespace* correspondente, desta forma, o banco de dados estará logicamente organizado.

Há ainda mais três estruturas que fazem parte da estrutura lógica do BD Oracle, que são os blocos de dados, os *extents* e os segmentos.

Os blocos de dados são o menor nível de granulosidade de um banco de dados Oracle e correspondem a um número específico de *bytes* no disco. Todas as operações de leitura, escrita, alocação de espaço em disco e alocação de partes da memória são feitas baseadas no tamanho do bloco.

O tamanho do bloco é especificado na criação do banco de dados e não pode mais ser alterado. Para alterar o tamanho do bloco é necessário recriar o banco de dados. Para questões de performance, o ideal é que o bloco de dados seja múltiplo do tamanho do bloco do sistema operacional.

O tamanho do bloco, não é totalmente destinado ao armazenamento das informações, sendo parte dele destinado ao armazenamento das estruturas de gerenciamento do próprio bloco. Essas estruturas são chamadas de *overhead* e fazem parte dela:

- *Header* – informações genéricas sobre o bloco, como o seu endereço e o tipo de objeto que armazena (tabelas, índices, outros). Seu crescimento é de cima para baixo;
- Diretório das tabelas – contém informações sobre as tabelas que têm linhas no bloco;
- Diretório de linhas – contém informações sobre os registros que estão no bloco.

A região do meio do bloco é um espaço livre destinado ao crescimento do bloco. O espaço livre é ocupado para novas inserções de registros ou para as atualizações dos registros atuais que requeiram um tamanho maior de armazenamento. Inicialmente, este espaço é um número contíguo de *bytes*, no entanto, registros apagados e atualizações constantes podem fragmentar esta área [ORO 2000].

A última parte do bloco de dados é a região destinada ao armazenamento das informações dos objetos. O crescimento desta área é de baixo para cima.

A figura 5.1 [ORO 2000] mostra o formato de um bloco de dados Oracle.

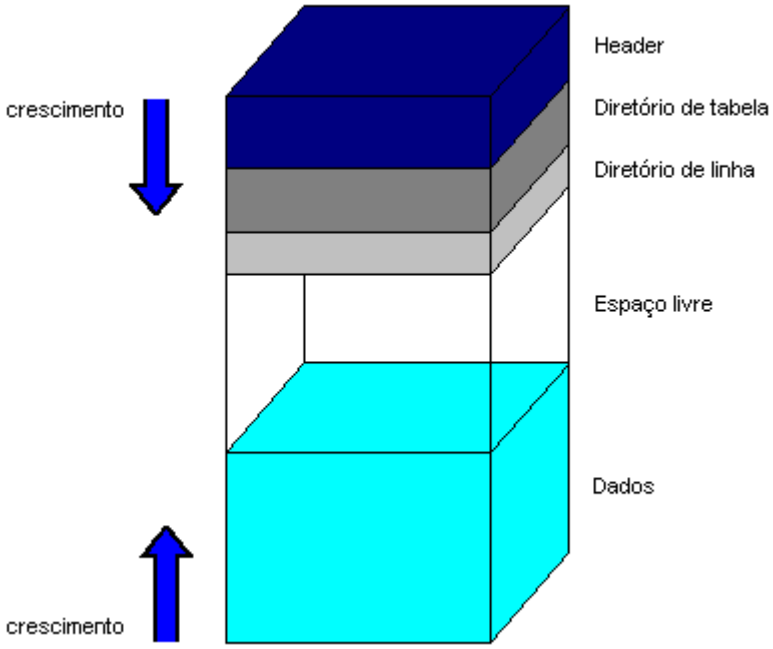

FIGURA 5.1 – Formato do bloco de dados Oracle

O próximo nível dentro da estrutura lógica de um banco de dados Oracle é o *extent*. Um *extent* é um número contíguo de blocos de dados.

Acima de *extent*, o próximo nível dentro da estrutura lógica do Oracle é o segmento, que nada mais é do que um conjunto de *extents*.

Os segmentos [ORO 2000, ORR 99] podem ser de vários tipos e os mais comuns são:

- tabela – armazenam os dados;
- índice – armazenam os índices;
- *rollback* – são usados pelas transações que realizam alterações nos dados para guardar os valores antigos dos dados alterados e são utilizados para desfazer as alterações;
- temporário – utilizados para realizar a organização das informações resultantes de consultas com as cláusulas *order by*, *group by*, *distinct*, entre outras.

Por exemplo, quando uma tabela é criada, o Oracle aloca um segmento com um *extent* inicial e este, por sua vez, com um conjunto de blocos de dados. Quando os blocos do *extent* estiverem cheios e mais espaço é necessário, o Oracle aloca um novo *extent* para este segmento.

A figura 5.2 representa a hierarquia de armazenamento do banco de dados Oracle, juntamente com a ligação entre a estrutura lógica e a física.

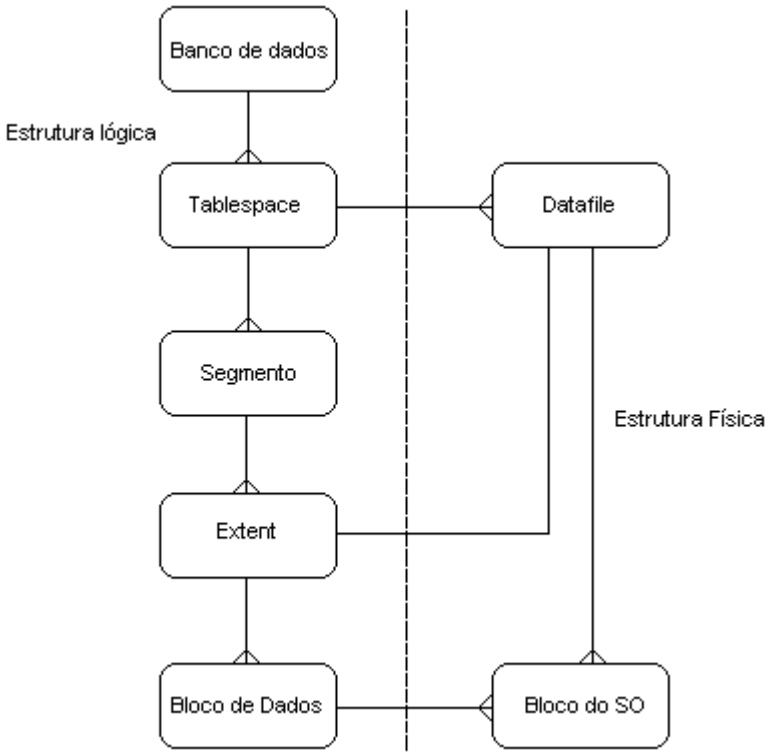

FIGURA 5.2 – Hierarquia de armazenamento do Banco de Dados Oracle

Analisando a figura 5.2 [ORO 2000] é possível observar que:

- um banco de dados é agrupado logicamente em *tablespaces*;
- um *tablespace* pode conter um ou mais segmentos;
- quando um segmento é criado, ele contém no mínimo um *extent*, que é um conjunto contíguo de blocos. Conforme o segmento cresce, novos *extents* são adicionados a ele;
- um bloco de dados Oracle é a menor unidade manipulável pelo Oracle.

5.2 Estrutura da Memória

O Oracle cria e mantém diversas estruturas em memória para a execução de várias tarefas, visando permitir com que muitos usuários acessem o banco de dados concorrentemente e possibilitando a alta performance exigida por um banco de dados [BOE 97]. Basicamente, esta memória é utilizada para armazenar o código dos programas que estão sendo executados e compartilhar dados entre os usuários e é chamada de *System Global Area* (SGA).

A *System Global Area* é uma parte da memória principal do computador que é compartilhada e contém informações dos sistemas e dados de controle para uma instância Oracle. O tamanho da SGA é definido em um arquivo de inicialização das instâncias. Uma

instância Oracle é composta pela SGA e por processos que executam em segundo plano responsáveis pelo funcionamento correto do atendimento às requisições dos usuários.

Cada vez que a instância é inicializada, o Oracle aloca a SGA na memória do servidor e esta área somente é desalocada quando a instância é finalizada. Cada instância possui sua própria SGA.

A *System Global Area* é composta por três estruturas [ORO 2000]:

- *Database buffer* – usado para armazenar os mais recentes dados solicitados pelos usuários;
- *Redo log buffer* – usado para registrar as alterações feitas no banco de dados que são posteriormente gravados nos arquivos de *redo*, discutidos no item 5.1.1, para casos de recuperação do BD;
- *Shared pool* – usada para armazenar os últimos comandos SQL submetidos ao banco de dados pelos usuários e, também, para armazenar o dicionário de dados da instância. A *library cache* é a estrutura que armazena, juntamente com os comandos submetidos ao banco, as árvores de análise (*parse tree*) e os planos de execução dos comandos. A *dictionary cache* é a estrutura que armazena o dicionário de dados.

Os processos em segundo plano em uma instância executam funções necessárias para atender as requisições de usuários concorrentes, sem comprometer a integridade e performance do sistema como um todo. Cada instância pode ter vários processos rodando em segundo plano, dependendo da configuração, mas cinco processos são básicos e que estão presentes em qualquer configuração.

Na tabela 5.1 estão apresentados os processos básicos que rodam em segundo plano em qualquer instância Oracle.

TABELA 5.1 – Processos básicos que rodam em segundo plano em uma instância Oracle

Processo	**Função**
Database Writer (DBWR)	Responsável por escrever os dados modificados na memória para o disco.
Log Writer (LGWR)	Responsável por gravar os dados do *redo log* da memória para o disco.
System Monitor (SMON)	Primeiro processo que é inicializado, verifica a integridade e inicia a recuperação do banco se for necessário.
Process Monitor (PMON)	Libera os recursos presos por processos que falham.
Checkpoint Process (CKPT)	Responsável por atualizar o banco de dados sempre que alterações forem feitas nos dados no disco.

5.3 Dicionário de Dados

O dicionário de dados [ORR 99, ORO 2000] é uma das partes mais importantes de um banco de dados Oracle e nada mais é do que um conjunto de tabelas e visões que são usadas como uma referência para fornecer informações sobre o banco de dados.

Através do dicionário de dados é possível obter as seguintes informações:

- nome dos usuários que acessam o banco de dados;
- os privilégios que cada usuário possui;
- nome dos objetos (tabelas, visões, índices, sinônimos, entre outros) de cada esquema do banco de dados;
- detalhes sobre as restrições de integridade do banco de dados;
- valores padrão definidas para colunas das tabelas;
- a quantidade de espaço físico ocupado pelos objetos;
- detalhes de auditoria.

O dicionário de dados é composto por duas partes:

- tabelas básicas - são tabelas que o servidor Oracle lê e escreve para manter o seu funcionamento. Dificilmente os usuários acessam estas tabelas pois suas informações são codificadas;
- visões - as visões decodificam e resumem as informações contidas nas tabelas básicas. São criados sinônimos públicos para que os usuários possam acessá-las mais facilmente.

As tabelas e visões do dicionário de dados são criadas durante a criação da instância e nunca devem ser manipuladas diretamente por comandos DML para não comprometer a integridade do banco de dados. No entanto, tudo o que o DBA precisa saber sobre a situação do banco de dados é conseguido através de consultas sobre as tabelas e visões do dicionário de dados.

As visões do dicionário de dados são divididas em três categorias [ORO 2000] e em muitos casos as visões podem conter informações similares e podem ser distinguidas umas das outras através de seus prefixos. As categorias são:

- prefixo USER - estas visões são acessíveis por qualquer usuário e geralmente se referem aos objetos que pertencem aos próprios usuários. Por exemplo, a visão USER_TABLES contém informações sobre todas as tabelas do usuário;
- prefixo ALL - estas visões são acessíveis por qualquer usuário e normalmente incluem a coluna OWNER. Elas retornam informações sobre os objetos que cada usuário tem acesso através de privilégios que são concedidos;
- prefixo DBA - as visões com o prefixo DBA fornecem informações sobre todos os objetos do banco de dados e também fornecem uma visão geral de todo o banco de dados. Estas visões só podem ser acessadas por usuários pelo administrador do banco ou por usuários aos quais for dado este privilégio.

A figura 5.3 ilustra a relação de hierarquia entre as três categorias de visões do dicionário de dados do BD Oracle.

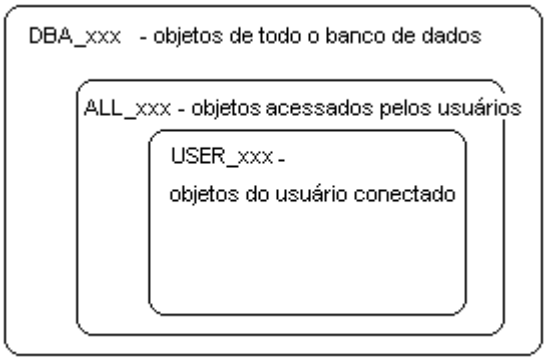

FIGURA 5.3 – Hierarquia das visões do dicionário de dados

Num banco de dados Oracle, há ainda um outro conjunto de visões que são importantes para que o administrador saiba como está o andamento do banco de dados. São as visões do desempenho dinâmico, que possuem este nome porque elas estão continuamente sendo atualizadas enquanto o banco de dados está aberto e em uso. Através delas é possível obter informações como, por exemplo, a situação das estruturas internas de disco e memória.

As visões de desempenho dinâmico [ORO 2000, SAR 2000] são identificadas pelo prefixo V_$. Os sinônimos públicos dessas visões têm o prefixo V$. Os administradores do banco de dados ou os usuários somente devem acessar os objetos V$, e não os objetos V_$.

Na tabela 5.2 são listadas algumas visões de desempenho dinâmico.

TABELA 5.2 – Visões de desempenho dinâmico

Visões	**Descrição**
V$PARAMETER	Contém informações sobre o arquivo de inicialização da instância.
V$SGA	Contém informações resumidas sobre a *System Global Area*.
V$SESSION	Lista as sessões atuais conectadas ao BD.
V$INSTANCE	Lista informações sobre a instância.
V$CONTROLFILE	Contém informações sobre os arquivos de controle.
V$DATABASE	Contém informações genéricas sobre o banco de dados.
V$DATAFILE	Contém informações sobre os *datafiles.*

5.4 Conclusões

O SGBD Oracle é um software extremamente complexo. No entanto, pode ser configurado em praticamente todos os seus aspectos, o que dá bastante flexibilidade aos administradores, mas torna a sua responsabilidade muito maior já que qualquer intervenção precipitada ou inadequada pode resultar em conseqüências desastrosas.

O capítulo 5 tratou do banco de dados Oracle. Inicialmente foram apresentados os aspectos básicos sobre o banco de dados, necessários para a compreensão de sua estrutura. O último item a ser discutido, apresentou o dicionário de dados cuja importância é fundamental para a implementação do protótipo, pois é sobre o dicionário de dados que o protótipo retirará as informações necessárias para o administrador efetuar os ajustes necessários sobre as instâncias.

6 A Ferramenta

Este capítulo apresenta e detalha o protótipo implementado neste trabalho. O seu desenvolvimento busca comprovar que é possível desenvolver uma ferramenta para resolver os problemas indicados no capítulo 3 e que possua a arquitetura apresentada no mesmo capítulo.

A ferramenta foi implementada utilizando a linguagem Java 2 e a tecnologia JSP versão 1.1. Na peça fundamental do protótipo, o servidor Web, foi utilizado o Tomcat versão 3.2.3. O principal banco de dados a ser administrado pela ferramenta é o Oracle *Enterprise Edition* versão 8.1.7.0.0 e o banco de dados secundário a ser acessado pela ferramenta é o SQL *Server* versão 2000. Todas estas ferramentas utilizaram o ambiente Windows 2000 *Professional*.

É importante lembrar que o servidor Web Tomcat é independente de plataforma e, portanto, poderia utilizar a plataforma Linux. No entanto, o fato de as demais ferramentas de desenvolvimento estarem disponíveis para a plataforma Windows, foi o fator decisivo para a escolha do sistema operacional.

Outro detalhe importante relativo às tecnologias utilizadas, mais precisamente sobre as versões de banco de dados: as que foram utilizadas, o foram devido a sua disponibilidade no momento do desenvolvimento. Mas, o Oracle a partir de sua versão 8.0.5 já poderia ser utilizado por conter os *drivers* JDBC que permitem a conexão da ferramenta. E, a Microsoft disponibiliza os *drivers* JDBC para o SQL *Server* a partir da versão 7.0 sendo apenas necessária a instalação do *driver* adequado à versão do banco.

Conforme foi mencionado no capítulo 3, uma boa ferramenta para administrar bancos de dados precisa auxiliar o administrador em suas tarefas diárias, como gerenciar o espaço ocupado pelas informações, avaliar o desempenho geral do BD, facilitar o gerenciamento dos principais tipos de objetos e permitir que o DBA exercite sua habilidade de administração. Esses são somente alguns dos tópicos que se espera de uma ferramenta de administração de BD e desta forma, a ferramenta desenvolvida também engloba estes tópicos.

A seguir são apresentadas as interfaces e as principais especificações das partes que compõem este protótipo.

6.1 Conexão com o banco de dados

De acordo com os estudos realizados no capítulo 4, item 4.2.2, a conexão com o banco de dados feita através da API JDBC, necessita do registro do *driver* adequado e da URL de conexão correta.

No protótipo desenvolvido, a conexão com o banco de dados ocorre percorrendo os seguintes passos:

1. preenchimento dos campos que irão compor a URL ;
2. acionamento do botão “Montar URL”;

3. escolha do banco de dados quando, internamente, o *driver* correspondente é registrado
4. acionamento do botão “OK”.

A figura 6.1 apresenta a interface para a conexão com o banco de dados.

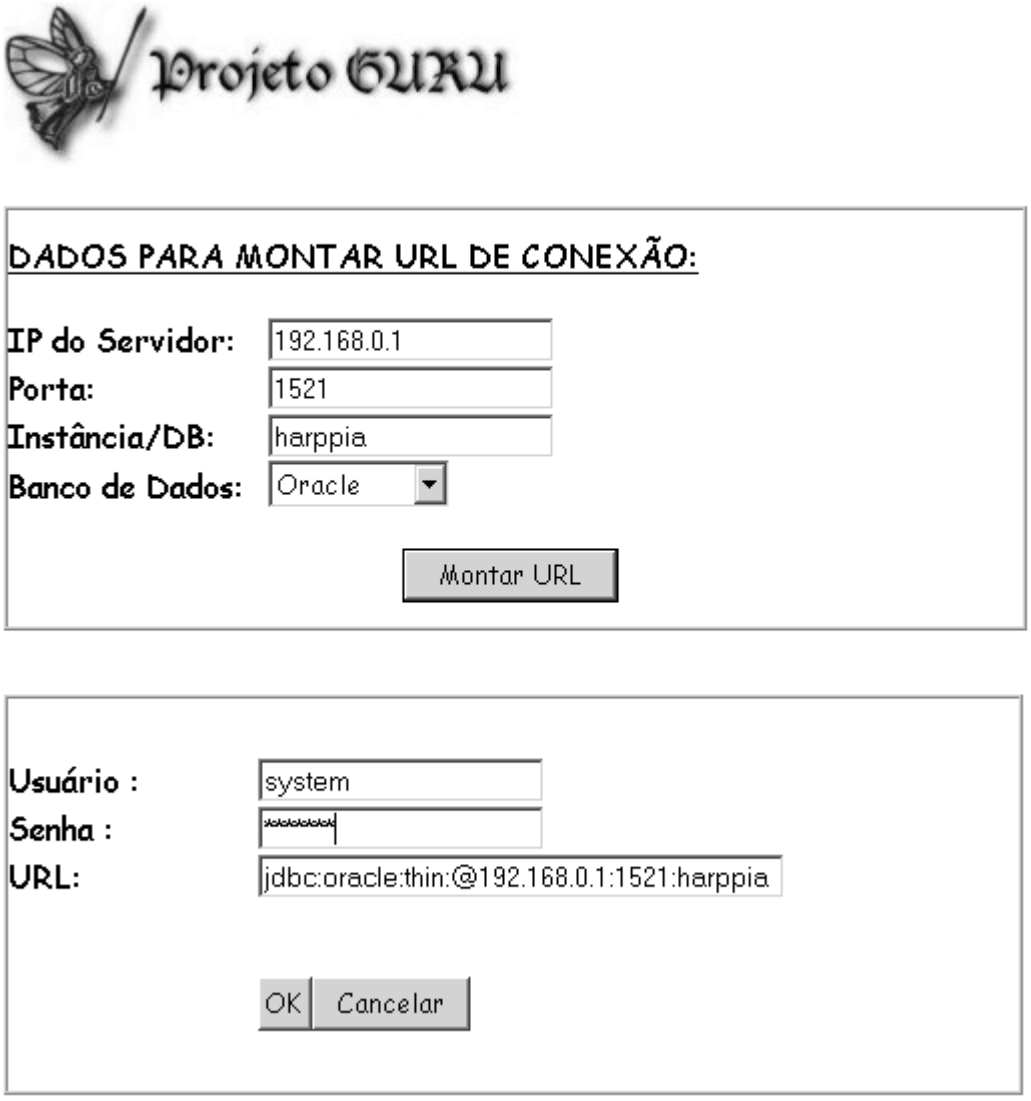

FIGURA 6.1 – Interface para a conexão com o banco de dados

Através da interface de conexão é possível conectar o banco de dados Oracle ou o SQL *Server*. Esta restrição foi definida apenas por questões de facilidade de digitação e disponibilidade de software. No entanto, a ferramenta está apta a conectar a outros bancos de dados fazendo pequenas alterações nesta interface de conexão. São elas:

- ao invés do usuário escolher o banco de dados, ele escolheria ou digitaria diretamente o *driver* do banco de dados;

- ao invés dos campos - IP do servidor, porta, instância ou BD - para preencher e montar a URL de conexão, estaria disponível somente a caixa de texto para que o usuário digitasse a URL no formato correto.

É importante lembrar que, para que a ferramenta conecte a qualquer banco de dados, é necessário indicar ao Tomcat onde se encontram os drivers JDBC do banco, definindo a variável de ambiente CLASSPATH.

Após conectar ao banco de dados desejado, a ferramenta apresenta a interface com opções de gerenciamento próprias de cada banco. A figura 6.2 apresenta as opções de gerenciamento do banco de dados Oracle. A figura 6.3 apresenta as opções de gerenciamento do banco de dados SQL *Server*.

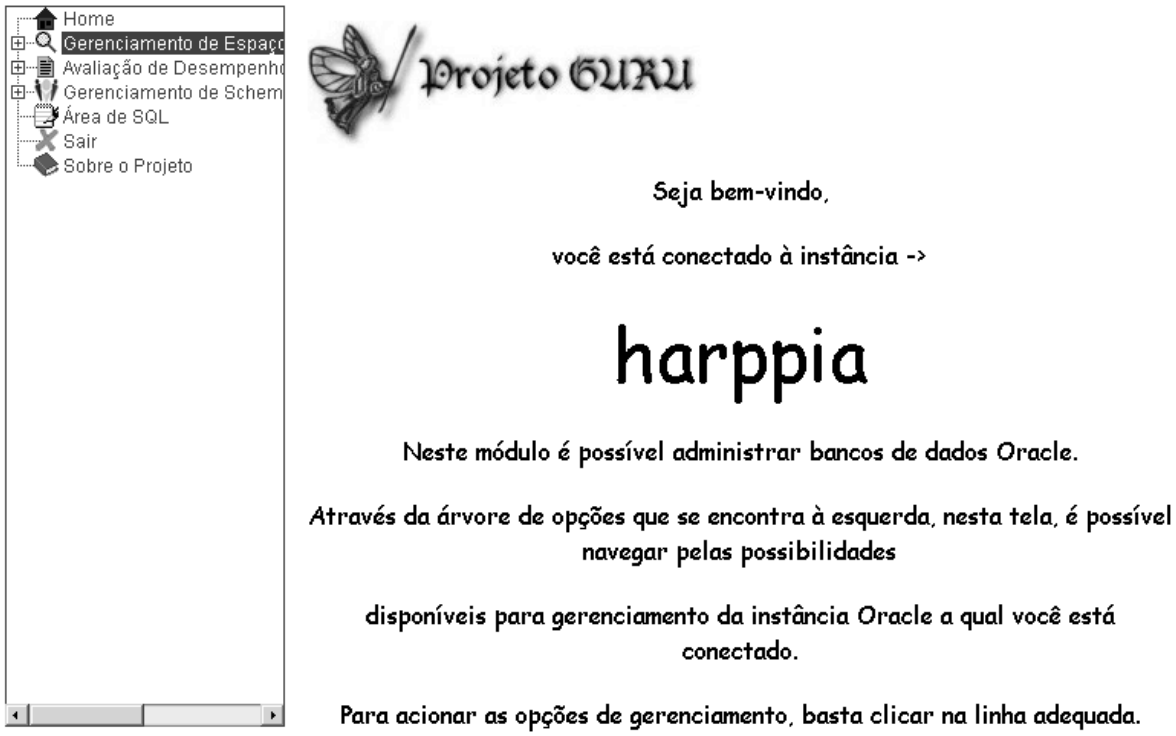

FIGURA 6.2 – Interface de gerenciamento do banco de dados Oracle

Ao fazer um comparativo entre as interfaces de gerenciamento de banco de dados Oracle e banco de dados SQL *Server*, figuras 6.2 e 6.3, respectivamente, é possível perceber a diferença de opções de gerenciamento entre os dois BD. Essa diferença se deve ao fato de que cada banco de dados possui estruturas particulares que diferem na forma de administrá-los e, portanto, exigem conhecimento específico. O Oracle foi escolhido como o principal banco de dados a ser administrado.

A inclusão do SQL *Server* foi feita com o intuito de comprovar que o protótipo pode ser utilizado para conectar-se a bancos de dados de diversos fornecedores. As opções de Gerenciamento de *Database* e Área de SQL foram possíveis por não exigirem conhecimentos específicos de administração, mas apenas comandos SQL ANSI 92.

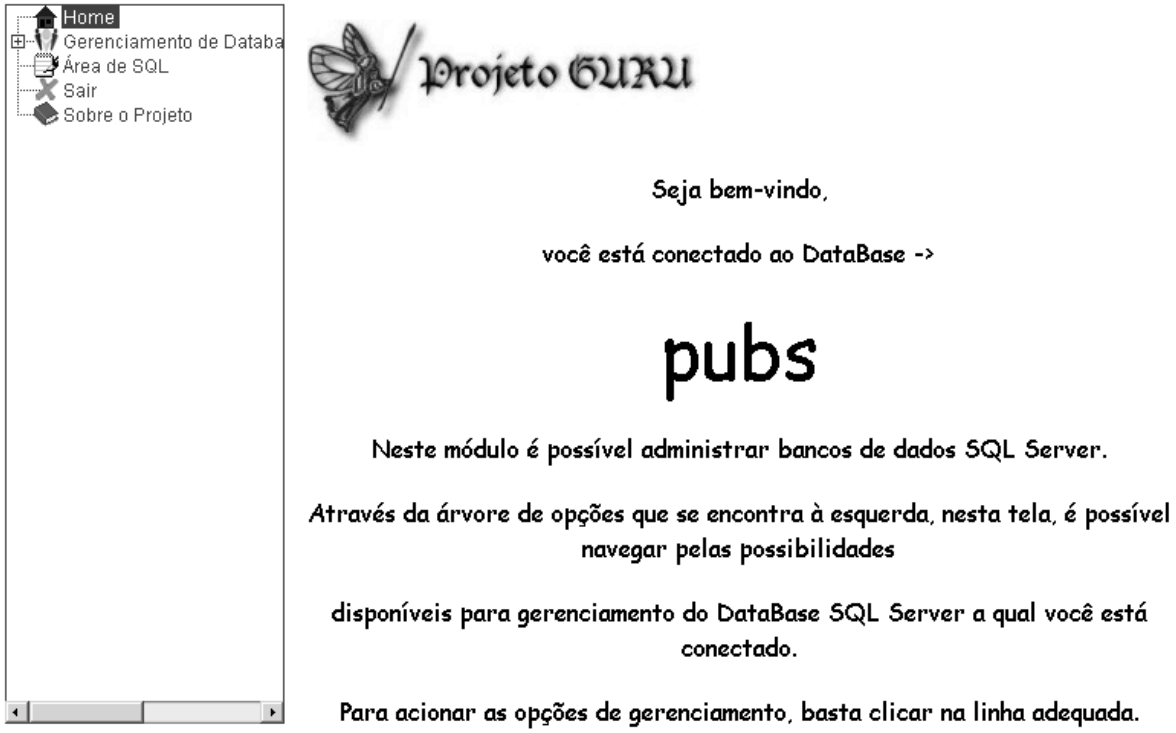

FIGURA 6.3 – Interface de gerenciamento do banco de dados SQL *Server*

Todas as opções de gerenciamento dos bancos de dados, tanto Oracle quanto SQL *Server*, quando acionadas apresentam inicialmente uma tela descritiva, chamada interface inicial. Esta é composta por um resumo com a utilidade da opção escolhida, uma breve descrição dos campos que a próxima interface possui e um *link* "INICIAR" que, ao ser acionado, direciona o foco da ferramenta para a interface de operação, onde o administrador poderá parametrizar a consulta ou visualizar o resultado da opção.

A seguir serão apresentadas as opções de gerenciamento que o protótipo permite ao banco de dados Oracle e ao SQL *Server*, quando for o caso.

6.2 Gerenciamento de espaço

Gerenciar o armazenamento dos objetos e suas informações é uma atividade que representa uma grande carga no trabalho do administrador. Inicialmente, o gerenciamento do espaço utilizado pelos dados no banco de dados pode parecer simplesmente uma tarefa de manutenção, mas ele influencia diretamente a operação geral do banco de dados. Por este motivo, a ferramenta implementada não poderia deixar de fora um aspecto que é tão

importante para o trabalho do DBA. Esta opção de gerenciamento está disponível somente para o banco de dados Oracle.

Na figura 6.4 são apresentadas as opções que o administrador tem disponível dentro do módulo de gerenciamento de espaço, para banco de dados Oracle: localização dos objetos com os *extents* utilizados e o *maxextents*; tabelas que não possuem mais espaço para crescer; tamanho médio da linha; tamanho máximo da linha; marca d'água; aglutinação dos espaços dos *tablespaces*.

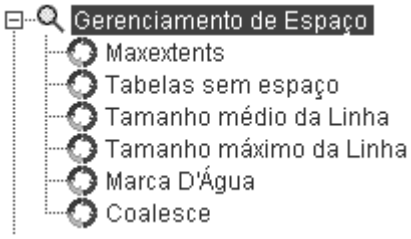

FIGURA 6.4 – Opções da interface de gerenciamento de espaço para BD Oracle

6.2.1 Maxextents

Ao selecionar a opção "*Maxextents*", a interface inicial é apresentada. A figura 6.5 mostra a interface inicial da opção "*Maxextents*".

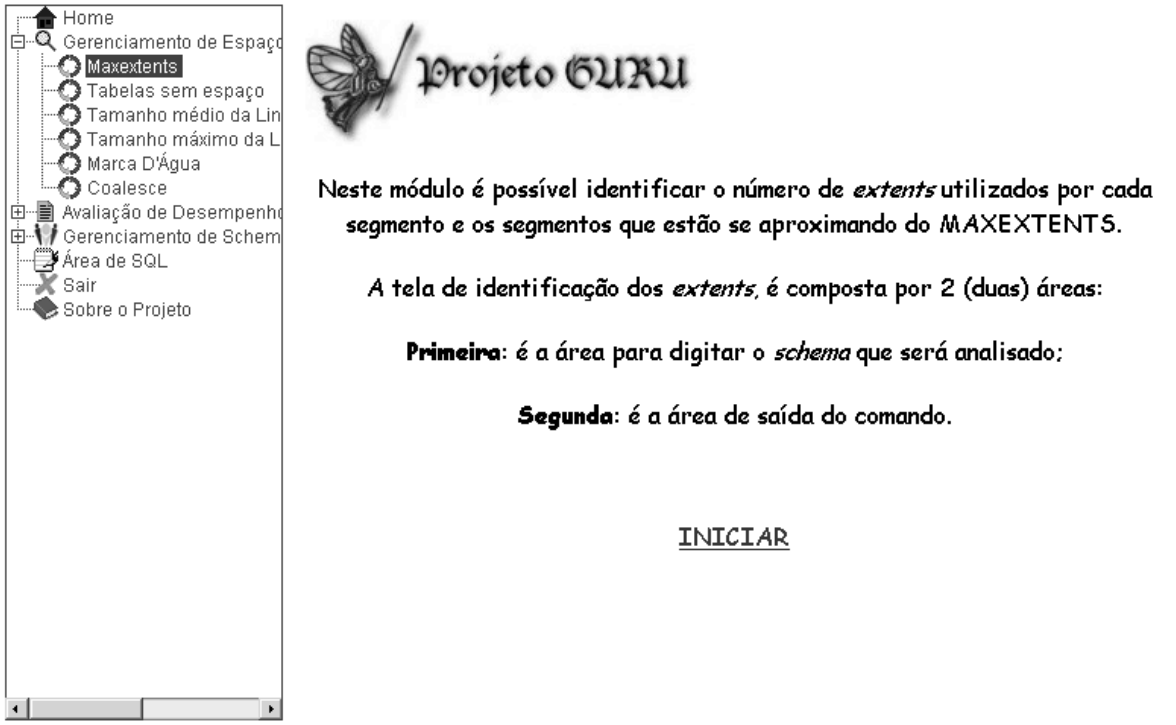

FIGURA 6.5 – Interface inicial da opção "*Maxextents*"

Na figura 6.5 foi acionado o *link* "INICIAR" e imediatamente a interface de operação, figura 6.6, foi apresentada. A interface de operação possui os seguintes atributos:

Schema, "Listar Segmentos" e área de saída do comando. O atributo *Schema* deve ser preenchido pelo DBA com o nome *schema* que ele deseja analisar e, após o preenchimento deste atributo, e ao ser acionado o botão "Listar Segmentos", a ferramenta apresenta os resultados dentro da área de saída.

O resultado do comando é a relação de objetos do *schema*, o número de *extents* utilizados, o número máximo de *extents* que o objeto pode utilizar e também é mostrado um " * " para chamar a atenção do objeto que está próximo de atingir o máximo de *extents* definidos para ele. Para fins de gerenciamento de espaço, esta consulta é uma excelente fonte de informação.

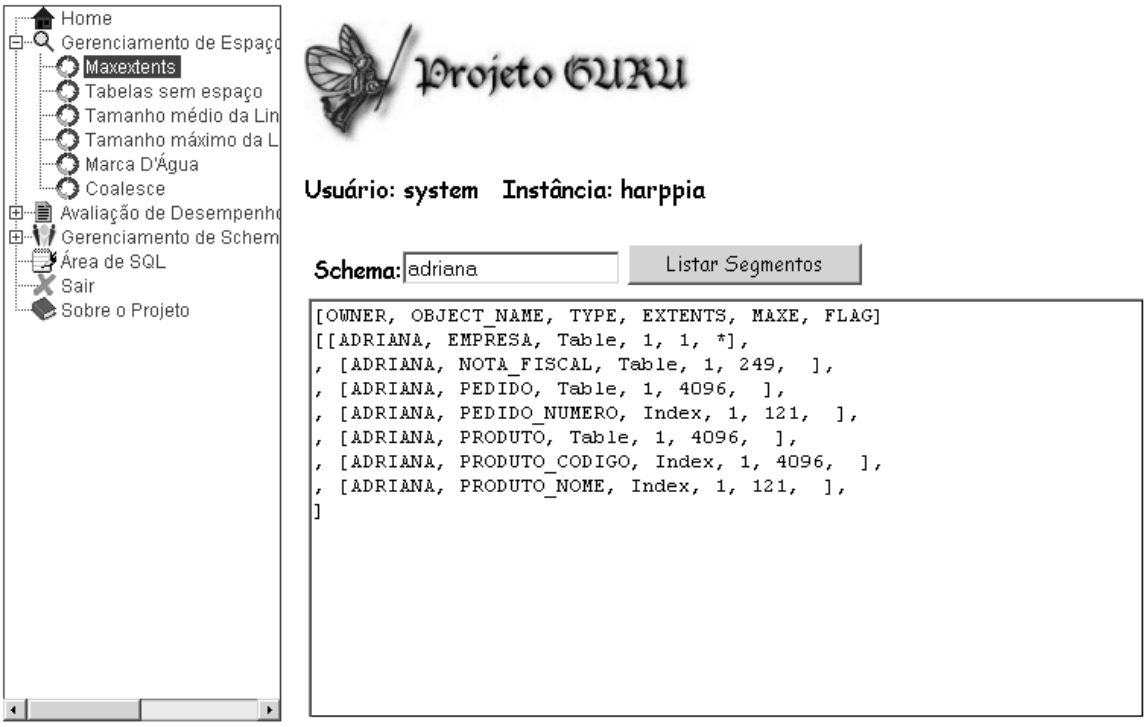

FIGURA 6.6 – Interface de operação da opção "*Maxextents*"

É importante salientar que neste item 6.2.1 foi apresentada e descrita a interface inicial com o objetivo de familiarizar o leitor com o funcionamento da ferramenta, mas nos próximos itens não serão abordadas as interfaces iniciais, uma vez que todas possuem a mesma estrutura e forma de funcionamento.

6.2.2 Tabelas sem espaço

A opção "Tabelas sem espaço", permite que o administrador tenha uma previsão de tabelas que não têm espaço livre suficiente para acomodar o próximo *extent* dentro do *tablespace*, por falta de espaço no *tablespace*. Através desta consulta o DBA consegue

evitar que o usuário final receba um erro e por conseqüência paralise o trabalho até que o erro seja corrigido.

A consulta de tabelas que não conseguem mais crescer é feita em toda a instância onde a ferramenta está conectada, o que significa que ao acionar o *link* “INICIAR” da interface inicial, a ferramenta já traz a interface de operação com o resultado da pesquisa em todo o banco de dados.

6.2.3 Tamanho médio da linha

O cálculo do tamanho médio de uma linha da tabela é importante no momento que o administrador irá definir os valores para os parâmetros de ocupação do bloco Oracle. O tamanho médio da linha influencia diretamente no parâmetro PCTFREE que especifica a quantidade de espaço livre que o Oracle reserva em um bloco de dados para acomodar a expansão das linhas armazenadas no momento em que elas são atualizadas. Outro parâmetro influenciado pelo tamanho médio da linha é o PCTUSED que é usado para classificar o bloco como disponível para inserção de novas linhas. Uma melhor definição sobre os parâmetros que definem um bloco Oracle foi feita no capítulo 5, item 5.1.2.

Na interface de operação o DBA deve informar o *schema* onde está a tabela que ele deseja consultar e o nome da tabela, para posteriormente acionar o botão “Calcular Tamanho Médio da Linha” para gerar o cálculo do tamanho médio da linha.

Certamente, em bancos de dados que estão em produção nas organizações, os *schemas* possuem inúmeras tabelas o que dificulta que o administrador conheça todas as tabelas de todos os *schemas*. Assim, nesta interface foi adicionado o botão “Listar Tabelas” e uma área de saída de comando abaixo deste botão. Este conjunto composto pelo botão e pela área de saída sob ele, é uma espécie de facilitador para o DBA, pois ao acionar o botão, é gerada uma listagem, figura 6.7, de todas as tabelas do *schema* informado.

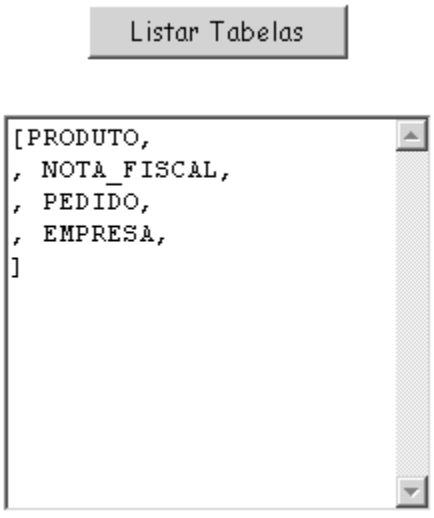

FIGURA 6.7 – Listagem das tabelas do *schema* informado

6.2.4 Tamanho máximo da linha

O tamanho máximo da linha é importante quando o administrador necessita tratar o encadeamento das linhas que causam uma *overhead* maior de E/S, degradando o desempenho geral do banco de dados.

O funcionamento da interface de operação desta opção é exatamente igual à opção tratada no item 6.2.3. A diferença é o procedimento interno da ferramenta que agora traz o tamanho máximo da linha da tabela indicada pelo administrador, ao invés do tamanho médio, que foi abordado no item 6.2.3.

6.2.5 Marca D'água

A informação referente à marca d'água de uma tabela é importante para que o administrador recupere o espaço não utilizado desta tabela. A visualização da marca d'água de uma tabela somente é possível após a análise da tabela, de forma que o dicionário de dados seja preenchido com os valores atualizados.

Na interface de operação da opção "Marca D'água", o administrador necessita informar o *schema* e a tabela para que seja possível consultar a informação que é mostrada quando o botão "Marca D'água" for acionado. Além de visualizar a marca d'água, a ferramenta possibilita que o administrador realize a análise da tabela para coletar as informações atualizadas, através do botão "Analisar Tabela".

Da mesma forma que na opção "Tamanho médio da linha", descrita no item 6.2.3, há um facilitador que lista as tabelas do atributo *schema* definido, nesta opção também há. Para visualizar o funcionamento do facilitador, ver figura 6.7.

6.2.6 Coalesce

Um dos maiores problemas que o DBA enfrenta na administração de banco de dados, dentro do enfoque do gerenciamento de espaço, é a questão da fragmentação do espaço livre dentro dos *tablespaces*. A ferramenta se propõe a resolver o problema da fragmentação do tipo *honeycomb*, onde os *extents* livres ficam adjacentes uns aos outros, através da opção "*Coalesce*".

A interface de operação apresenta o atributo "*Tablespace*" que deve ser preenchido com o nome do *tablespace* que deve ser aglutinado. Após o preenchimento deste atributo, a aglutinação é realizada através do acionamento do botão "Aglutinar *Tablespace*". Na área de saída do comando é exibido o resultado da operação de aglutinação que informa ao administrador o sucesso ou fracasso do procedimento. Uma causa de fracasso pode ser, por exemplo, a digitação de um valor inválido para o atributo *tablespace*. Ainda, nesta interface de operação, há um facilitador que ajuda o administrador a visualizar todos os *tablespaces* criados na instância na qual a ferramenta está conectada. A listagem é exibida através do acionamento do botão "Listar *Tablespaces*".

6.3 Avaliação de desempenho

O desempenho do banco de dados determina efetivamente a sua disponibilidade, isto é, de nada adianta o BD estar em operação se cada tarefa submetida levar muito tempo para retornar o resultado. Um sistema mal ajustado pode ser considerado a ruína para o DBA. Portanto, para a atividade de ajuste, ele precisa ter em mãos uma ferramenta que lhe mostre o caminho a seguir para melhorar o desempenho do BD. Por este motivo, a ferramenta desenvolvida possui um módulo para auxiliar o administrador a encontrar possíveis problemas de desempenho.

Na implementação atual, o módulo de avaliação de desempenho está disponível somente para Oracle, pois todas as consultas são realizadas sobre o dicionário de dados específico do Oracle.

O fato de as consultas, do módulo de "Avaliação de Desempenho", extraírem as informações das visões de desempenho dinâmico do dicionário de dados, significa que para que elas sejam significativas é importante executar as consultas apenas depois que o banco de dados estiver operacional durante algum tempo e houver dados de desempenho suficientes acumulados para formar uma boa base. Maiores informações sobre as visões de desempenho dinâmico podem ser obtidas no capítulo 5, item 5.3.

As seguintes grandes áreas do banco de dados, sob o ponto de vista do desempenho, são abordadas neste módulo e apresentadas na figura 6.8:

- *System Global Area*;
- área utilizada para classificação de informações;
- armazenamento de informações;
- identificação dos comandos que estão causando maior *overhead* do sistema.

FIGURA 6.8 – Opções do módulo Avaliação de Desempenho

6.3.1 *System Global Area* – SGA

A *System Global Area* é uma área da memória crítica para a operação de qualquer instância Oracle. Sob o ponto de vista do ajuste de desempenho, o cache do *buffer* de dados e o *pool* compartilhado são duas áreas de ajuste mais importantes dentro da SGA.

A seguir será discutido como a ferramenta desenvolvida trata estes dois enfoques.

6.3.1.1 *Database Buffer Cache*

Os *buffers* de dados são estruturas de memória da SGA que contém os blocos de dados Oracle usados mais recentemente. Como o Oracle pode acessar os blocos de dados em cache diretamente dos *buffers* de dados da memória, em vez de ter que ler esses dados do disco, menos E/S de disco precisa ser executada e o desempenho melhora.

A medida mais importante do desempenho do *buffer* de dados é a taxa de *hits* do *buffer* de dados, que mede o número de vezes em que o Oracle encontrou os blocos de dados que ele precisava na memória, ao invés de ler esses dados do disco.

Ao acionar o *link* "INICIAR" da interface inicial, a ferramenta já traz a interface de operação, figura 6.9, com o resultado da consulta.

A interface de operação da opção "*Database Buffer Cache*" é dividida em três áreas:

- primeira - taxa de *hits* do *buffer* de dados ;
- segunda - avaliação realizada sobre a taxa que indica a situação do *buffer* de dados;
- terceira - dicas que orientam as ações do administrador baseadas na avaliação efetuada sobre a taxa.

É importante observar o detalhe da figura 6.9 onde está a segunda área da interface. Ela pode apresentar-se em três cores diferentes, onde cada cor possui um significado diferente:

- verde - quando o valor da taxa de *hits* está excelente, isto é, o banco de dados está conseguindo buscar a maioria das informações solicitadas na memória, sendo o acesso ao disco muito pequeno;
- amarelo - quando o valor da taxa de *hits* está dentro de um padrão aceitável, ou seja, muitas informações solicitadas precisaram ser buscadas do disco. Nestes casos o administrador deve ficar alerta;
- vermelho - quando o valor da taxa de *hits* está abaixo de valores aceitáveis, o que significa que o banco de dados está indo buscar as informações no disco mais vezes do que ele as encontra na memória. Necessita intervenção do administrador.

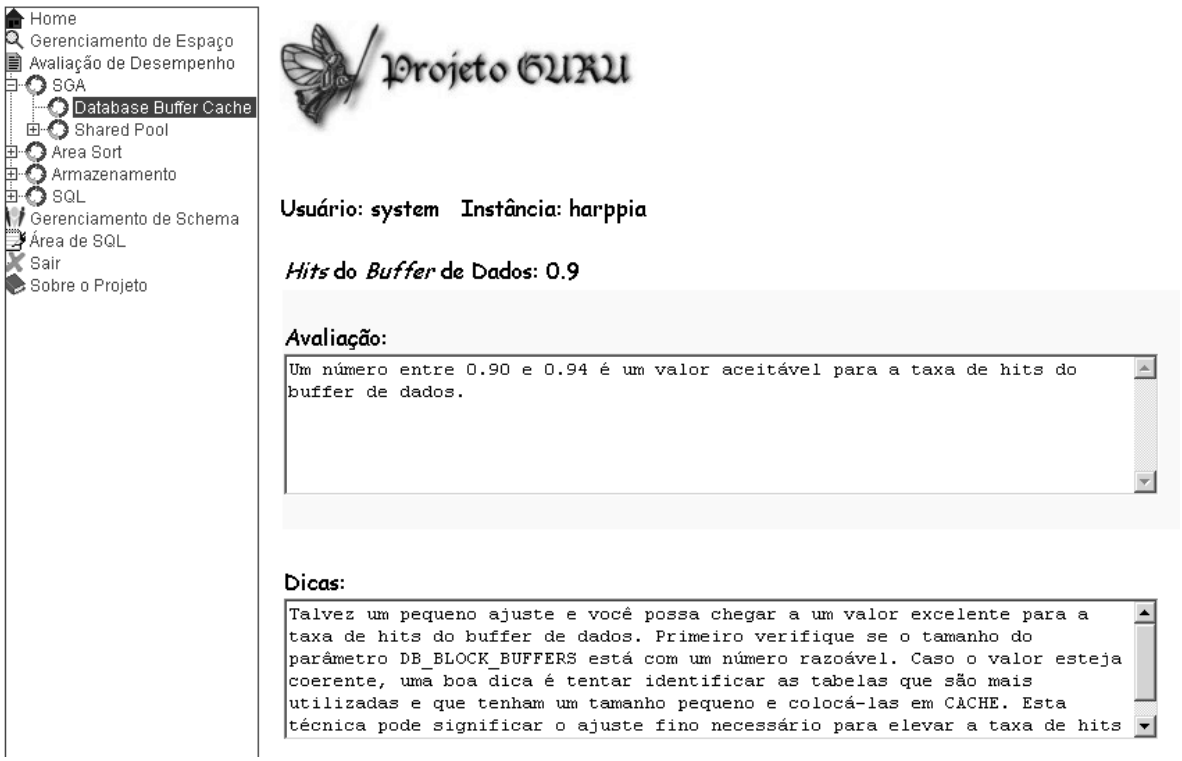

FIGURA 6.9 – Interface de Operação da opção "*Database Buffer Cache*"

6.3.1.2 *Shared Pool*

A ferramenta permite que o administrador avalie o estado da *shared pool* através das opções de consultas a seguir:

Library Cache

O funcionamento desta opção é exatamente igual à opção tratada no item 6.3.1.1, isto é, ao acionar o *link* "INICIAR" da interface inicial, a ferramenta já traz a interface de operação com o resultado da consulta. A diferença é o procedimento interno da ferramenta que agora traz a taxa de *hits* do cache de biblioteca.

A opção "*Library Cache*", apresenta uma área com o valor da taxa de *hits*, uma área com a avaliação da taxa e uma com dicas úteis relacionadas com a avaliação efetuada.

Dicionário de Dados

O funcionamento da opção "Dicionário de Dados" também é exatamente igual à discutida no item 6.3.1.1, diferenciando apenas o procedimento interno da ferramenta que agora traz a taxa de *hits* do dicionário de dados.

Nesta opção, a interface de operação também está dividida em três áreas: taxa, avaliação e dicas.

6.3.2 *Area Sort*

O banco de dados Oracle precisa de espaço para classificar os dados quando são executadas operações como, por exemplo, a criação de índices ou consultas com cláusulas *ORDER BY*. O ajuste das operações de classificação melhora significativamente o desempenho geral do banco de dados e não somente das consultas que classificam informações.

O módulo de "*Area Sort*" permite que o administrador avalie o desempenho da área utilizada para classificação das informações, visualizando o desempenho através da relação entre as classificações em memória e classificações em disco, além de permitir a identificação dos usuários que estão mal definidos, ou seja, que estão utilizando áreas não temporárias de disco para efetuar as classificações.

6.3.2.1 *Sort* Memória *X Sort* Disco

Sempre que o banco de dados executar uma classificação, ele primeiro tenta executá-la na memória. Se houver memória suficiente, então a classificação é executada na memória. Caso contrário, o Oracle classifica os dados em disco no *tablespace* especificado como padrão para o usuário.

A opção "*Sort* Memória *X Sort* Disco" permite que o administrador avalie a relação entre classificações efetuadas em memória em relação às efetuadas em disco.

Após acionar o *link* da interface inicial, imediatamente, a interface de operação, traz o resultado. A interface de operação é dividida em 3 áreas: taxas, avaliação e dicas. A área das taxas, figura 6.10, é composta pelas quantidades de classificações em memória, classificações em disco e o percentual de classificações em disco. O atributo "avaliação" funciona como um alerta para o administrador, pois ele troca a cor de fundo de acordo com a porcentagem de classificações em disco.

FIGURA 6.10 – Taxas de classificações da opção "*Sort* Memória X *Sort* Disco"

6.3.2.2 Definição de Usuários

Quando as classificações são efetuadas em disco, é melhor que elas utilizem *tablespaces* temporários, pois estes são otimizados durante a sua utilização em classificações de informações em disco. A vinculação do *tablespace* temporário é a nível de usuário e sua definição é feita durante a criação dos usuários, podendo ser alterada a qualquer momento.

A opção "Definição de Usuários" permite ao administrador identificar usuários mal definidos no banco de dado, isto é, usuários que estão utilizando *tablespaces* permanentes para realizar as classificações em disco.

A interface de operação com os resultados da consulta são mostrados ao administrador após o acionamento do link na interface inicial. Na interface de operação há duas áreas:

- usuários – relação de todos os usuários definidos no BD, seus respectivos *tablespaces* e o tipo do *tablespace*;
- dicas – com orientações aos administradores.

6.3.3 Armazenamento

A maneira como as informações estão sendo armazenadas pode ter um impacto sobre o desempenho do banco de dados. O armazenamento influi diretamente o desempenho, principalmente em leituras grandes e seqüenciais, por exemplo. Para manter o banco de dados com todo o desempenho possível é essencial ter um armazenamento ajustado.

Dentro do aspecto de armazenamento, para obter o máximo de desempenho, os aspectos que a ferramenta abordará são: encadeamento de linhas no banco de dados, encadeamento de linhas em tabelas específicas e gerenciamento dos *extents*.

6.3.3.1 Encadeamento de Linhas BD

O encadeamento de linhas ocorre quando uma linha é grande demais para caber em um bloco de dados Oracle e, sempre que possível, deve ser evitado porque ele causa uma sobrecarga de E/S, levando à redução do desempenho.

A opção “Encadeamento de Linhas BD” da ferramenta, permite que o administrador tenha uma visão geral de todo o banco de dados em relação à existência de linhas encadeadas. Esta consulta é útil para que o administrador faça uma avaliação rápida da seriedade do problema no banco de dados.

A interface de operação possui três atributos – número de linhas encadeadas no BD, avaliação e dicas – que são exibidos assim que o *link* da interface inicial for acionado. O atributo “avaliação ” funciona da mesma forma que o apresentado na figura 6.9, isto é, ele troca a cor de fundo operando como um alerta visual da situação do encadeamento de linhas no banco de dados.

6.3.3.2 Encadeamento de Linhas Tabelas

Após verificar que há linhas encadeadas no banco de dados através da opção vista no item 6.3.3.1, o administrador precisa identificar precisamente em quais tabelas está ocorrendo o encadeamento. A opção “Encadeamento de Linhas Tabelas” permite que o administrador realize esta verificação. A interface de operação possui os atributos descritos na tabela 6.1.

TABELA 6.1 – Atributos da opção “Encadeamento de Linhas Tabelas”

Atributos	Tipo	Descrição
Schema	Texto	*Schema* onde está a tabela a ser analisada
Tabela	Texto	Nome da tabela que será analisada
Analisar Tabela	Botão	Ao ser acionado, atualiza a contagem de linhas encadeadas
Contar Linhas Encadeadas	Botão	Ao ser acionado, exibe o número de linhas encadeadas
Saída	Texto	Onde é exibido o número de linhas encadeadas
Listar Tabelas	Botão	Lista todas as tabelas do *schema* informado
Listagem	Texto	Onde são listadas as tabelas do *schema*

6.3.3.3 *Extents*

A questão do crescimento dos *extents* é bastante discutível, porém existem fortes evidências que a redução do crescimento dos *extents* pode melhorar o desempenho do banco de dados. A opção "*Extents*" da ferramenta permite que o administrador verifique o número de *extents* que os objetos estão ocupando. A interface de operação possui os atributos descritos na tabela 6.2.

TABELA 6.2 – Atributos da opção "*Extents*"

Atributos	Tipo	Descrição
Schema	Texto	*Schema* onde estão os objetos que serão analisados
Objeto	Texto	Nome do objeto que será analisado
Extents Objeto	Botão	Ao ser acionado, faz a contagem dos *extents* do objeto informado no atributo "Objeto"
Extents Tabelas	Botão	Ao ser acionado, exibe o número de *extents* de todas as tabelas do *schema*
Extents Índices	Botão	Ao ser acionado, exibe o número de *extents* de todos os índices do *schema*
Saída	Texto	Onde são exibidos os *extents*
Listar Tabelas	Botão	Lista todas as tabelas do *schema* informado
Listar Índices	Botão	Lista todos os índices do *schema* informado
Listagem	Texto	Onde são listadas as tabelas ou os índices do *schema*

6.3.4 SQL

A escrita e o ajuste das declarações SQL não fazem parte do trabalho do administrador do banco de dados. No entanto, ele deve saber, dentre as declarações que estão sendo executadas, quais são as mais caras em termos de consumo de memória e CPU.

A ferramenta oferece ao administrador opções de consulta para facilitar a localização das declarações SQL que mais estão consumindo recursos do servidor. A seguir serão abordadas estas opções.

6.3.4.1 Maior Consumo de Disco

Esta opção permite ao administrador identificar as declarações mais caras em termos de disco, isto é, que estão custando mais em termos de E/S.

Ao acionar o *link* da interface inicial, a interface de operação é chamada, onde o administrador precisa informar um valor mínimo de acesso a disco, através do preenchimento do atributo “Leituras em Disco”. Esta restrição é necessária, caso contrário, a consulta se tornaria ineficaz, pois traria todas as declarações executadas pelo banco de dados, o que não faria sentido do ponto de vista de identificar as consultas mais caras. Caso o administrador queira verificar todas as declarações executas, então basta deixar um valor “0” no atributo “Leituras em Disco”.

Na área de saída da consulta, existem cinco colunas de informações, que serão descritas a seguir:

- SQL_TEXT - declaração SQL;
- DISK_READS - número de acesso a disco que a declaração consumiu;
- EXECUTIONS - número de vezes que a declaração foi executada;
- RATIO - relação entre o número de acessos a disco e número de vezes que a declaração foi executada;
- USERNAME - nome do usuário que analisou pela primeira vez a declaração.

6.3.4.2 Maior Consumo de CPU

As declarações que estão consumindo mais memória e, por conseqüência, mais CPU, podem ser vistas na opção “Maior Consumo de CPU”.

O funcionamento desta opção é igual ao apresentada no item 6.3.4.1. No entanto, a coluna DISK_READS que existe na opção do item 6.3.4.1, aqui chama-se BUFFER_GETS e indica o número de leituras feitas na memória para a determinada declaração SQL.

6.4 Gerenciamento de *Schema*

O módulo de gerenciamento de *schema* permite que o administrador do banco de dados acesse rapidamente os vários tipos de objetos que pertencem aos diferentes *schemas* da instância à qual a ferramenta está conectada.

O módulo de gerenciamento de *schema* está disponível para Oracle e para SQL *Server*, onde é chamado de gerenciamento de *database*.

Dentre os vários tipos de objetos que os bancos de dados possuem, a ferramenta gerencia os principais. Para bancos de dados Oracle: índices, tabelas e visões. Para SQL *Server* somente tabelas. Na figura 6.11 são apresentadas as opções que o administrador têm disponível dentro do módulo de gerenciamento de *schema*, para banco de dados Oracle e na figura 6.12 são apresentadas as opções que o administrador possui para gerenciar *database* em bancos de dados SQL *Server*.

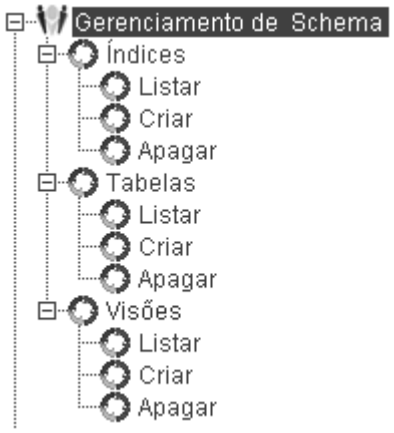

FIGURA 6.11 – Opções de Gerenciamento de *Schema* para bancos de dados Oracle

FIGURA 6.12 – Opções de Gerenciamento de *Database* para BD SQL *Server*

A seguir serão apresentadas as opções de gerenciamento de *schema* para banco de dados Oracle, que foi escolhido para ser detalhado em função de haver maiores opções e porque o gerenciamento de *database* para SQL *Server* possui a mesma forma de funcionamento.

6.4.1 Índices

Os índices são objetos importantes do ponto de vista de desempenho e acessá-los de maneira rápida, é de fundamental importância para que o DBA realize um gerenciamento ágil. No módulo de manipulação dos índices é possível listar, criar e apagar índices.

6.4.1.1 Listar

Na opção para listar índices é possível escolher entre a listagem de todos os atributos que descrevem os índices ou selecionar os principais atributos de descrição dos índices. Após acionar o *link* da interface inicial, a interface de operação é acionada para que os atributos sejam preenchidos. A interface de operação é dividida em quatro áreas, descritas na tabela 6.3.

TABELA 6.3 – Áreas da opção para listar índices

Área	Tipo	Descrição
Primeira	Texto	*Schema* ao qual pertencem os índices
Segunda	*CheckBox*	Atributos dos índices
Terceira	Texto	Área para exibição do comando a ser executado
Quarta	Texto	Saída da execução do comando

Entre a primeira e segunda área, há o botão “Montar SQL” que, ao ser acionado, irá montar o comando na terceira área, para ser executado. A montagem do comando pode ser feita de duas formas:

- acionando diretamente o botão “Montar SQL” o comando montado está pronto para listar todos os atributos que descrevem os índices;
- marcando algum atributo da segunda área e acionando o botão “Montar SQL”, o comando será montado somente para listar os atributos marcados.

Quando o comando é montado na terceira área, o administrador ainda pode alterá-lo de acordo com a necessidade, já que esta área é editável. A figura 6.13 mostra a terceira área da opção “Listar” índices do *schema* informado, com a opção de listar todos os atributos. Para executar o comando que foi montado na terceira área é necessário acionar o botão “Executar”.

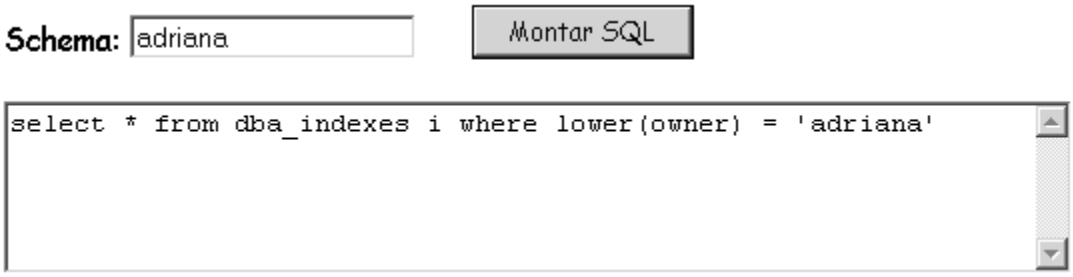

FIGURA 6.13 – Terceira área da opção “Listar” índices mostrando todos os atributos

6.4.1.2 Criar

O módulo para criar índices permite que o administrador crie índices para o *schema* indicado, de uma forma rápida e fácil, uma vez que a composição do comando que cria os índices é feita através do preenchimento dos principais campos.

Os atributos da interface de operação da opção que cria índices são:

- *Schema* - atributo do tipo texto que indica em qual *schema* o índice será criado;
- Campo - atributo do tipo texto que indica sobre qual campo o índice será criado;
- Tabela - atributo do tipo texto que indica de qual tabela é o campo que será indexado;
- *Tablespace* - atributo do tipo texto que indica em qual *tablespace* ficará armazenado o índice;
- Montar SQL - botão que ao ser acionado monta o comando que irá criar o índice;
- Comando - atributo do tipo texto, onde será exibido o comando;
- Executar - botão que ao ser acionado, cria o índice;
- Resultado - saída do comando que foi executado indicando sucesso ou fracasso;
- Listar Tabelas - botão que lista todas as tabelas do *schema* e que funciona como ajuda no momento de criar o índice;
- Listar Campos da Tabela - botão que lista todos os campos da tabela que foi indicada no atributo "Tabela" e também funciona como ajuda para criar o índice;
- Listar *Tablespaces* - botão que lista todos os *tablespaces* que o *schema*, onde o índice será criado, possui quotas e, também, funciona como ajuda para criar o índice;
- Listagem - atributo do tipo texto onde as ajudas são exibidas.

A montagem do comando que será executado no atributo "Comando" é feita através do preenchimento dos campos que ficam acima do botão "Montar SQL", conforme pode ser visto na figura 6.14. No entanto, o único atributo cujo preenchimento é obrigatório é o atributo "*Schema*". Caso os demais não sejam preenchidos, a ferramenta irá montar o comando com palavras-chave, conforme ilustra a figura 6.15.

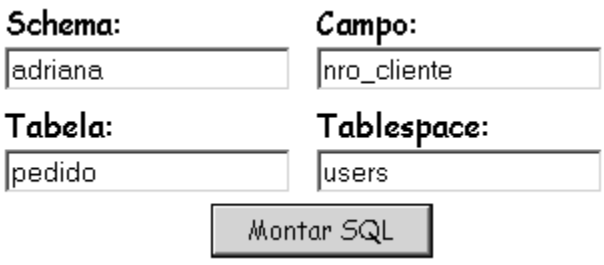

FIGURA 6.14 – Campos que montam o comando da opção "Criar" índices

```
create unique index adriana.<NOME_DO_INDICE>
on adriana.<nome_da_tabela>(<nome_do_campo>)
pctfree 10 initrans 2 maxtrans 255 tablespace
<nome_da_tablespace>
storage ( initial 10240 next 10240 minextents 1 maxextents 121
pctincrease 50)
```

FIGURA 6.15 – Comando montado com palavras-chave

6.4.1.3 Apagar

Na opção para apagar índices, o administrador pode rapidamente localizar e apagar um índice de um determinado *schema*. A interface de operação da opção "Apagar" índices é composta pelos atributos apresentados na tabela 6.4:

TABELA 6.4 – Atributos da opção "Apagar" índices

Atributo	Tipo	Descrição
Schema	Texto	*Schema* ao qual pertence o índice que será apagado
Tabela	Texto	Tabela a qual pertence o índice que será apagado
Índice	Texto	Nome do índice que será apagado
Montar SQL	Botão	Monta o comando para apagar o índice
Comando	Texto	Local onde será exibido o comando a ser executado
Executar	Botão	Executa o comando exposto no atributo "Comando"
Saída	Texto	Onde é exibido o resultado do comando executado e indica sucesso ou fracasso
Listar Tabelas	Botão	Lista todas as tabelas do *schema* informado
Listar Índices	Botão	Lista todos os índices do *schema* informado
Listagem	Texto	Onde são listadas as tabelas ou os índices do *schema*

O preenchimento do atributo "*Schema*" é obrigatório. Já o preenchimento do campo "Índice" é opcional, isto é, se ele não for preenchido o comando para apagar índice é montado com a palavra-chave <nome_do_indice> que deverá ser substituída pelo nome correto do índice que o administrador deseja apagar.

O atributo "Tabela", pode ser considerado como uma ajuda para que o administrador localize facilmente o nome do índice a ser apagado, já que, normalmente, ele sabe a qual tabela pertence o índice, mas desconhece seu nome. Desta forma, a operação se torna fácil: basta informar o nome da tabela no atributo correspondente, acionar o botão "Listar Índices", procurar o nome do índice a ser apagado no atributo "Listagem" e informá-lo no atributo "Índice" ou substituí-lo diretamente no atributo "Comando".

6.4.2 Tabelas

Dentro do módulo de gerenciamento de *schema* há a opção de listar, criar e apagar tabelas de forma rápida e fácil. De maneira geral, o funcionamento da manipulação de tabelas é muito semelhante ao visto anteriormente no item 6.4.1 para a manipulação de índices.

O gerenciamento de tabelas está disponível para bancos de dados Oracle e SQL *Server*, sendo que o modo de operação, nos dois bancos diferentes, é o mesmo.

6.4.2.1 Listar

A opção para listar tabelas é composta por duas opções de consulta:

- listar todos os campos que descrevem as tabelas;
- selecionar, entre os principais campos, alguns mais significativos;

Para montar o comando que irá listar as tabelas, o administrador precisa informar o nome do *schema* para o qual ele deseja listar as tabelas. Portanto, o atributo "*schema*" da interface de operação da opção "Listar" tabelas é obrigatório.

Para bancos de dados SQL *Server*, não existe o atributo "*Schema*" e a listagem das tabelas ocorre para todo o *database* no qual a ferramenta está conectada.

6.4.2.2 Criar

A opção para criar tabelas em bancos de dados Oracle possui o atributo "*Schema*" que deve ser preenchido para que a ação de criar tabela seja executada através da ferramenta.

Ao informar o valor do atributo *schema* e acionar o botão "Montar SQL" o comando padrão é montado no atributo "Comando". A partir do comando montado, o administrador substitui os valores padrão para os valores corretos. O comando padrão para bancos de dados Oracle está ilustrado na figura 6.16.

```
create table adriana.<NOME_DA_TABELA>
(<NOME_DO_CAMPO> <TIPO_DO_CAMPO>(<TAMANHO_DO_CAMPO>) )
pctfree 10 pctused 40 initrans 1 maxtrans 255
storage (initial 131072 next 16384 minextents 1 maxextents 249
pctincrease 50)
tablespace <nome_da_tablespace>
```

FIGURA 6.16 – Comando padrão para criar tabelas em BD Oracle

Na interface de operação, há, ainda, o atributo "*Tablespace*" que pode ser preenchido com o valor do *tablespace* onde a tabela deve ser criada. Caso este atributo seja preenchido, então o valor padrão `<nome_do_tablespace>` é substituído pelo valor correto, através do acionamento do botão "Montar SQL".

Como forma de auxiliar a tarefa de criar tabelas, existem os botões “Listar Tabelas” e “Listar *Tablespaces*” que listam, respectivamente, as tabelas que existem no *schema* e os *tablespaces* onde o *schema* possui quotas para criar tabelas.

Para bancos de dados SQL *Server*, o funcionamento é bastante semelhante. No entanto há duas diferenças básicas: a montagem do comando padrão é feita diretamente após o acionamento do botão “Montar SQL”, já que não existe o atributo “*Schema*” e não há o atributo “*Tablespace*” e, portanto, não há o botão “Listar *Tablespace*” na ajuda.

O comando padrão para criar tabelas no banco de dados SQL *Server* pode ser observado na figura 6.17.

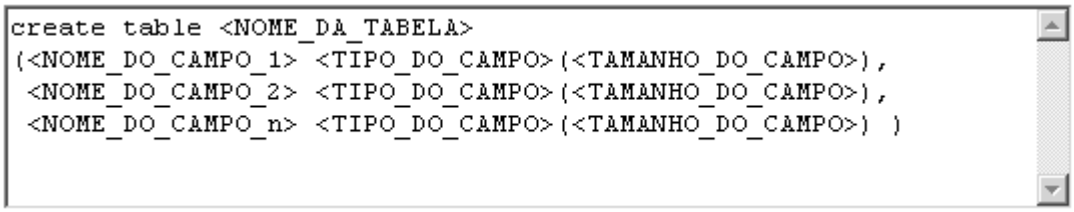

FIGURA 6.17 – Comando padrão para criar tabelas em BD SQL *Server*

6.4.2.3 Apagar

As interfaces de operação da opção “Apagar” tabelas para bancos de dados Oracle e SQL *Server* possuem os seguintes atributos descritos na tabela 6.5.

TABELA 6.5 – Atributos da opção “Apagar” tabelas para BD Oracle e SQL *Server*

Atributo	Tipo	BD	Descrição
Schema	Texto	Oracle	*Schema* ao qual pertence a tabela que será apagada
Tabela	Texto	Oracle e SQL Server	Nome da tabela que será apagada
Montar SQL	Botão	Oracle e SQL Server	Ao ser acionado monta o comando para apagar tabela
Comando	Texto	Oracle e SQL Server	Exibe o comando para apagar a tabela
Executar	Botão	Oracle e SQL Server	Ao ser acionado executa o comando que está no atributo “Comando”
Resultado	Texto	Oracle e SQL Server	Saída da execução do comando, indica sucesso ou fracasso
Listar Tabelas	Botão	Oracle e SQL Server	Ao ser acionado lista todas as tabelas do *schema* ou *database*
Listagem	Texto	Oracle e SQL Server	Relação das tabelas existentes

6.4.3 Visões

Como outra opção do módulo de gerenciamento de *schema*, há a opção para gerenciar visões. Dentro desta opção é possível listar, criar e apagar visões. A opção para manipular visões está disponível somente para bancos de dados Oracle.

6.4.3.1 Listar

Na opção para listar visões é possível escolher entre a listagem de todos os campos que descrevem as visões ou selecionar os principais campos de descrição.

Na interface de operação da opção "Listar" visões somente o atributo "*Schema*" é obrigatório e deve ser preenchido. Após o preenchimento, para realizar a listagem de todos os campos descritos das visões, basta acionar o botão "Montar SQL" e em seguida o botão "Executar". Caso o DBA escolha fazer a consulta selecionando os campos a serem listados, figura 6.18, então os campos do tipo *checkbox* devem ser preenchidos para posterior acionamento do botão "Montar SQL", seguido do botão "Executar".

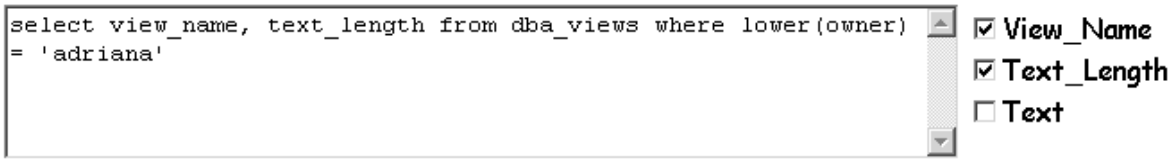

FIGURA 6.18 – Comando para listar visões selecionando os campos

6.4.3.2 Criar

A opção "Criar" dos objetos do tipo visão, permite que o administrador do banco de dados rapidamente crie visões. As visões, por definição, não contêm e nem realmente armazenam dados. Ao invés disso, elas "herdam" os dados das tabelas nas quais elas se baseiam.

Para criar visões através da ferramenta, o administrador somente precisa informar o *schema* onde ela será criada e acionar o botão "Montar SQL". Feito isto, o comando básico de criação de visão é montado:

`create view adriana.<NOME_DA_VISAO> as select * from adriana.<nome_da_tabela>`

Posteriormente, o DBA poderá alterar o comando montado para adequá-lo à sua necessidade.

Para facilitar a criação de visões, a ferramenta traz vários atributos que auxiliam a montagem do comando. São eles:

- Tabela - caso este atributo esteja preenchido, então o comando básico é montado com o nome da tabela correto ao invés do padrão **`<nome_da_tabela>;`**
- Listar Tabelas - botão que, ao ser acionado, traz uma listagem de todas as tabelas do *schema*;

- Listar Campos da Tabela - para utilizar este botão é necessário ter preenchido anteriormente o atributo “Tabela”, então a ferramenta traz uma listagem de todos os campos que a tabela informada possui.

6.4.3.3 Apagar

A interface de operação da opção para apagar visões possui os atributos apresentados na tabela 6.6:

TABELA 6.6 – Atributos da opção “Apagar” visões

Atributo	Tipo	Descrição
Schema	Texto	*Schema* ao qual pertence a visão que será apagada
Visão	Texto	Nome da visão que será apagada
Montar SQL	Botão	Monta o comando para apagar a visão
Comando	Texto	Local onde será exibido o comando a ser executado
Executar	Botão	Executa o comando exposto no atributo “Comando”
Saída	Texto	Onde é exibido o resultado do comando executado e indica sucesso ou fracasso
Listar Visões	Botão	Lista todas as visões do *schema* informado
Listagem	Texto	Onde são listadas as visões do *schema*

6.5 Área de SQL

Possuir uma área livre para execução de comandos SQL é uma qualidade imprescindível de uma boa ferramenta para administrar bancos de dados, pois é através dela que os problemas mais complexos ou específicos são resolvidos.

A área de SQL para bancos de dados Oracle aceita todos os tipos de comando para gerenciamento do banco de dados, dos usuários e gerenciamento de *schemas*. Apenas não é possível finalizar e inicializar as instâncias.

Para bancos de dados SQL *Server*, todos os comandos pertinentes à gerenciamento de *databases* podem ser executados na área de SQL. Na figura 6.19 está apresentada a interface de operação do módulo “Área de SQL” para banco de dados Oracle.

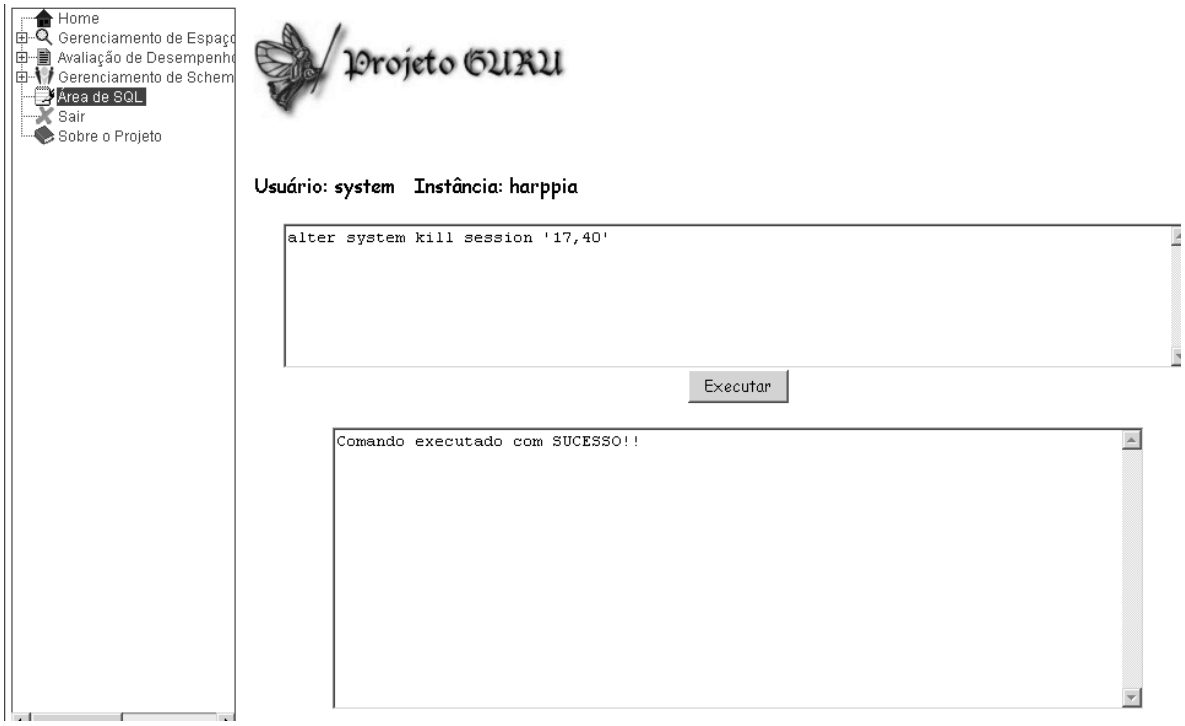

FIGURA 6.19 – Módulo “Área de SQL” para Oracle

O exemplo acima descreve o módulo de área livre para comandos SQL utilizado para finalizar uma sessão de usuário que estava conectado ao banco de dados, através do comando **`alter system kill session '17,40'`**. Após digitar o comando no atributo “Comando”, o botão “Executar” é acionado, o comando é executado e o resultado trazido no atributo “Resultado”.

O módulo “Área de SQL” para SQL *Server* possui exatamente os mesmos atributos e a mesma forma de funcionamento.

6.6 Conclusões

Este capítulo apresentou detalhadamente a ferramenta implementada, de forma a familiarizar o leitor com os termos e formas de funcionamento que são próprias da ferramenta.

Foram tratados os itens mais importantes que a ferramenta possui. São eles:

- modo como a ferramenta realiza a conexão ao banco de dados Oracle e ao SQL *Server*;
- forma como a ferramenta trata do problema de gerenciamento de espaço em bancos de dados Oracle;
- maneira como a ferramenta auxilia o administrador de um banco de dados Oracle a realizar uma avaliação constante do desempenho do BD;

- como é possível, utilizando a ferramenta, acessar de maneira rápida e fácil os diversos tipos de objetos que os bancos de dados Oracle e SQL *Server* possuem;
- modo como a ferramenta permite que os administradores de bancos de dados Oracle e SQL *Server* possam realizar os mais excêntricos tipos de comandos de gerenciamento destes bancos.

7 Estudo de Caso

O objetivo deste capítulo é descrever o estudo de caso elaborado neste trabalho para ser utilizado como base para aplicar e validar a ferramenta desenvolvida para administrar bancos de dados remotos.

7.1 Domínio

O estudo de caso utiliza como fonte de dados, uma instância de banco de dados Oracle de uma empresa distribuidora de produtos, cuja matriz localiza-se em Porto Alegre - RS. A base de dados engloba informações referentes aos produtos, distribuição, clientes, fornecedores, entre outras. Este domínio foi adotado para o estudo de caso por vários motivos relevantes ao enriquecimento da experiência:

- trata-se de uma base de dados Oracle com aproximadamente 2GB de informações;
- este banco de dados foi recentemente implantado, vindo de outra plataforma;
- a empresa possui acesso rápido à Internet e um *firewall* protegendo a rede interna.

O ambiente descrito acima é um excelente laboratório para comprovar a eficiência e eficácia da ferramenta desenvolvida. No entanto, a empresa solicitou que nenhum dado que possa identificar a empresa seja divulgado, por questões de segurança. Esta restrição não prejudicou o estudo de caso, pois foi feita a implantação e utilização da ferramenta em sua forma completa. Aqui estarão publicadas somente as interfaces neutras.

7.2 Implantação da Ferramenta

A instalação da ferramenta ocorreu em duas etapas:

- na primeira etapa foi feita a instalação do Tomcat e da ferramenta, em uma máquina com endereço IP fixo que pertence a rede interna da empresa. Este etapa foi realizada como parte do estudo de caso;
- na segunda etapa foram alteradas as configurações do *firewall* da empresa, para que todas as requisições que chegam ao endereço IP do servidor *Tomcat* sejam direcionadas corretamente a ele. Esta etapa foi executada pelo administrador da rede e não faz parte do escopo deste trabalho.

7.3 Atividades Desenvolvidas

Feita a instalação, os passos seguintes foram o acesso e, posteriormente, o gerenciamento do banco de dados através da ferramenta.

Todos os módulos de gerenciamento que a ferramenta possui foram testados. No entanto, a seguir serão descritos os mais relevantes e aqueles que não revelam informações que possam identificar a empresa.

Acesso à ferramenta e conexão com o BD

O acesso à ferramenta foi feito através de um navegador da máquina cliente, fora do ambiente da empresa, conectada à Internet, digitando o endereço do servidor *Tomcat*, como segue:

```
http://endereço_ip_do_Tomcat/guru/index.jsp
```

Para conectar a ferramenta ao banco de dados que foi analisado bastou informar os dados de conexão ao banco de dados da empresa na interface apropriada, como no exemplo da figura 6.1.

Avaliação de desempenho

O banco de dados sobre o qual foi desenvolvido este estudo de caso, foi implantado a menos de um ano e, por este motivo, torna-se uma excelente fonte para avaliação de desempenho. Embora durante todo o período de operação do banco de dados, da implantação até agora, várias devem ter sido as vezes em que foram feitas avaliações de desempenho, uma base de dados de um ano pode ser considerada "amadurecida". Assim sendo, é possível formular conclusões sobre possíveis alterações no banco de dados devido a falhas de desempenho, com maiores probabilidades de acerto.

Dentre as opções que o módulo de avaliação de desempenho que a ferramenta desenvolvida possui, foram analisados os seguintes aspectos:

- ***Database Buffer Cache***

Durante a verificação do *database buffer cache* foi possível observar que, neste aspecto e à primeira vista, o banco de dados parece estar bem ajustado. No entanto, lendo mais calmamente a "dica" oferecida pela ferramenta, nota-se que o índice de avaliação do *database buffer cache* está no limite para tornar-se problemático.

Com base nestas informações, o administrador toma consciência de que a avaliação constante deste item é fundamental, pois qualquer mudança de comportamento pode causar a queda de performance.

A figura 7.1 representa o desempenho do banco de dados em relação aos *buffers* de dados. A tarja preta sobre o nome da instância que é visto nesta figura, e nas demais, serve para garantir o sigilo que foi exigido pela empresa onde este estudo de caso foi realizado.

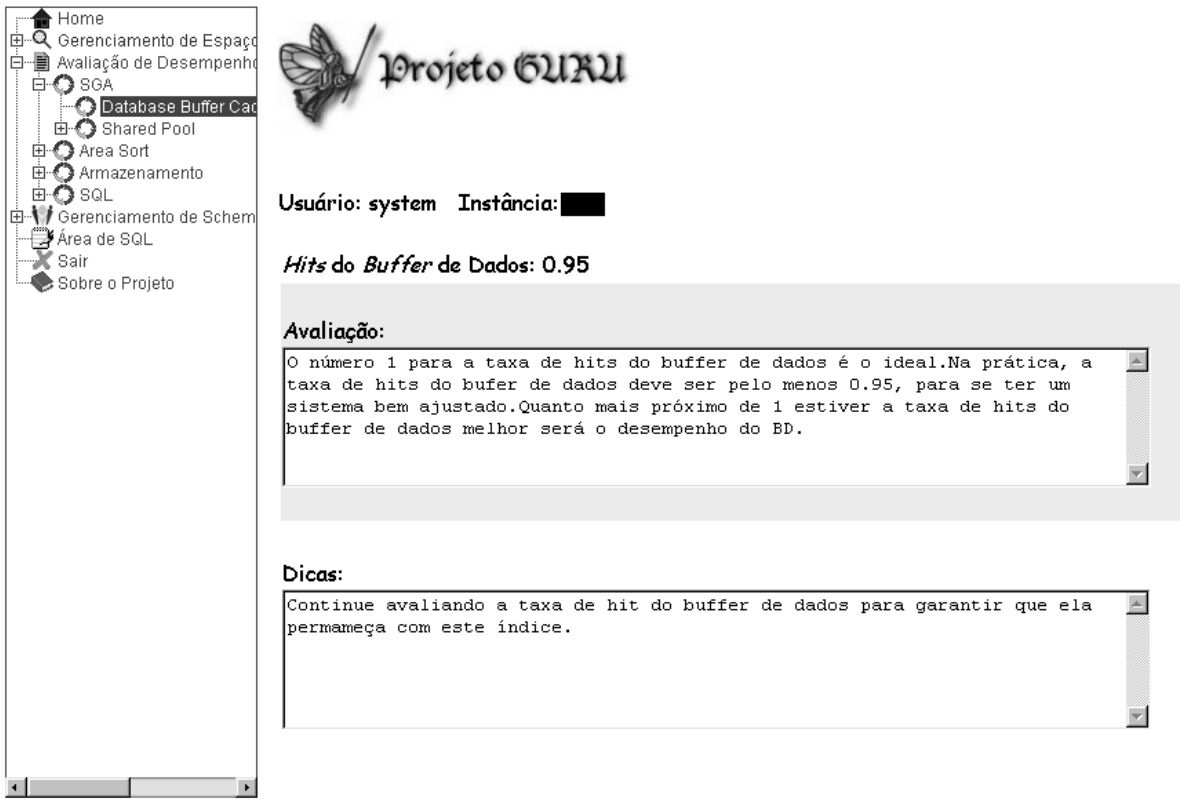

FIGURA 7.1 – *Database Buffer Cache*

- ***Library Cache:***

A observação da *Library Cache*, mostrou que neste aspecto o banco de dados está bem configurado. A figura 7.2 ilustra a consulta da *Library Cache*.

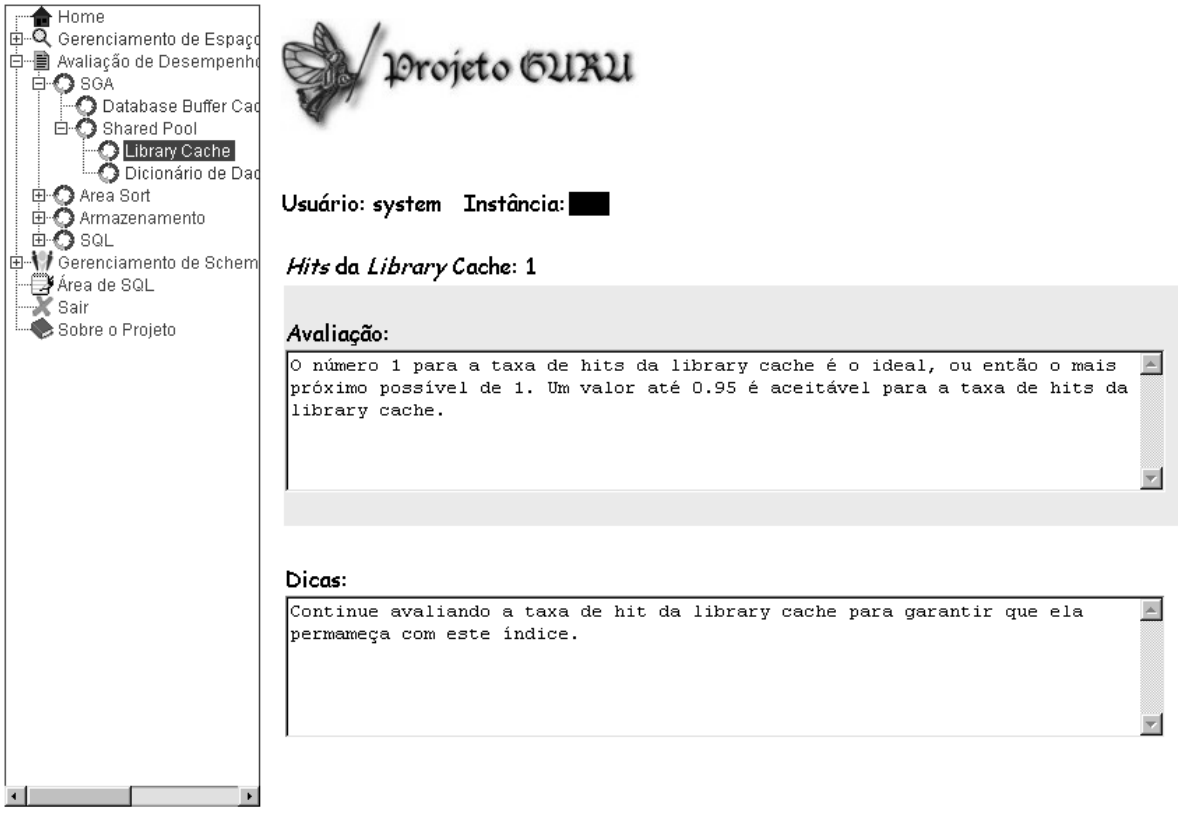

FIGURA 7.2 – *Library Cache*

- **Dicionário de Dados:**

Ao analisar o dicionário de dados, foi observado que este item está apresentando problemas, isto é, a taxa de *hits* está atingindo um ponto crítico e requer que o administrador faça uma avaliação minuciosa para verificar se é uma situação temporária ou se há necessidade de realizar uma intervenção. Esta situação pode ser vista na figura 7.3.

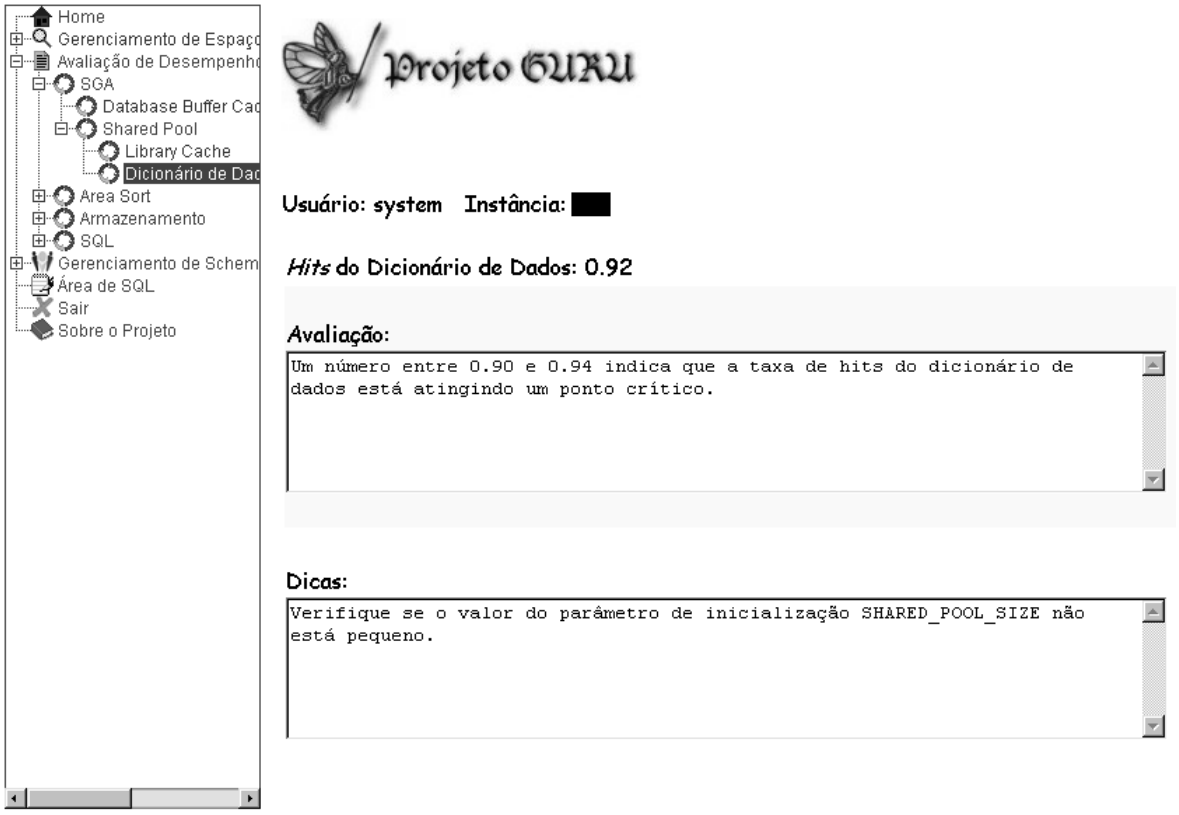

FIGURA 7.3 – Dicionário de Dados

Através das consultas de avaliação da SGA (*Database Buffer Cache*, *Library Cache* e Dicionário de Dados), foi possível observar que se faz necessário um pequeno ajuste para que a área do dicionário de dados se estabilize. Assim sendo, foi alterado o parâmetro SHARED_POOL_SIZE, conforme a "dica" oferecida pela ferramenta e, após finalização e reinicialização do banco de dados, uma nova consulta foi realizada no dicionário de dados. Seu *hit* aumentou para 0.98, o que já representou uma melhora significativa na performance geral do BD.

- ***Sort* Memória X *Sort* Disco**

Ao realizar a análise do desempenho das classificações de informações no banco de dados, foi obtida uma resposta positiva, pois o banco de dados está bem configurado neste aspecto, já que todas as classificações foram feitas em memória e nenhuma classificação em disco havia sido feita. Em um sistema bem ajustado esta é a situação ideal.

A figura 7.4 ilustra a consulta da ferramenta desenvolvida sobre as estatísticas das classificações das informações no banco de dados.

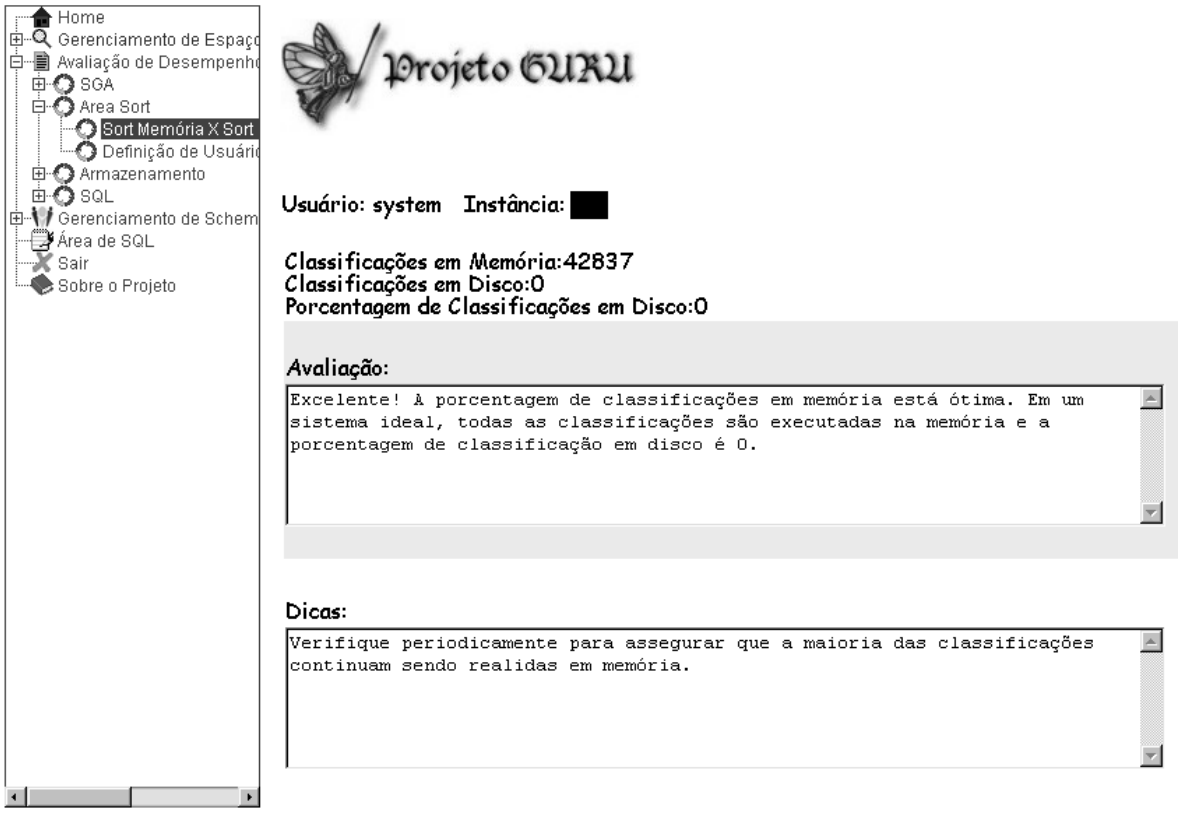

FIGURA 7.4 – Classificações das informações no banco de dados

- **Encadeamento de Linhas BD**

A análise do armazenamento do ponto de vista de desempenho mostrou-se muito interessante, pois foram observados aspectos que estavam prejudicando determinadas operações no banco de dados e que não estavam sendo identificadas.

Para executar esta etapa, foram seguidos os passos a seguir:

1. consulta das linhas encadeadas a nível de banco de dados - o início da investigação para descobrir linhas encadeadas iniciou-se com a verificação no nível de banco de dados, a fim de identificar a ocorrência do problema. Através do módulo “Encadeamento de Linhas BD”, figura 7.5, foi identificado que há a ocorrência de linhas encadeadas no banco de dados. Feita esta consulta, vale aqui a observação de que o número de linhas encadeadas que foram encontradas, ainda é pequeno. No entanto, se a maioria destas linhas pertencerem a uma única tabela e esta tabela fora importante para a operação do sistema, então o banco de dados está com um problema sério que deve ser resolvido;

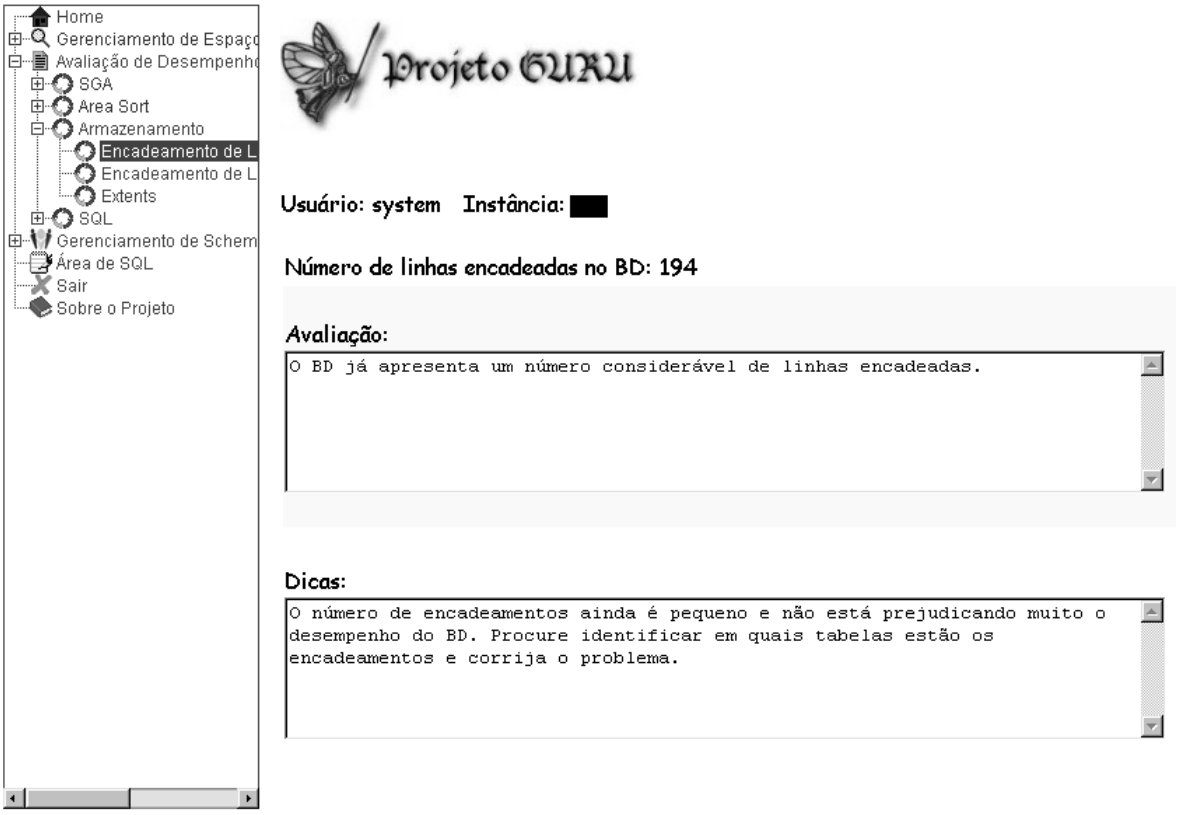

FIGURA 7.5 – Linhas encadeadas no banco de dados

2. identificação da tabela com problema - sabendo que há tabelas com linhas encadeadas no banco de dados, deve-se identificar em qual tabela está o problema. A procura pelas tabelas que estão com linhas encadeadas foi realizada através da opção “Encadeamento de Linhas Tabelas”, onde é necessário informar o *schema* e o nome da tabela. Após várias consultas, a tabela com problema foi identificada. Tratava-se da tabela que armazena os produtos que a empresa distribui. A consulta que identificou linhas encadeadas no banco de dados trouxe 194 linhas. Destas, 183 pertenciam à tabela de produtos e as outras 11 estavam distribuídas em 2 outras tabelas de menor importância para a operação do sistema.

3. solução - para solucionar o problema, foi utilizado o módulo “Área de SQL” onde foram executados uma série de procedimentos:

- identificação das linhas encadeadas através do comando

```
analyze tabela_produto list chained rows
into chained_rows;
```

- criação de uma tabela temporária para servir de *backup* das linhas encadeadas pelo comando

```
create table produto_temp as
select * from tabela_produto
where rowid in (select head_rowid
                from chained_rows);
```

- apagar as linhas encadeadas da tabela original com o comando

```
delete from tabela_produto
where rowid in (select head_rowid
                from chained_rows);
```

- inserir as linhas da tabela temporária para a original utilizando o comando

```
insert into tabela_produto
select * from produto_temp;
```

- apagar a tabela temporária através do comando

```
drop table tabela_produto.
```

Da mesma forma como os comandos acima foram executados para a tabela de produtos, as outras duas tabelas que também apresentavam linhas encadeadas, passaram pelos mesmos procedimentos.

Feito isto, uma nova verificação no módulo “Encadeamento de Linhas BD ” foi realizada, tendo sido identificado que o problema havia sido resolvido com sucesso.

7.4 Conclusões

Este capítulo serviu para mostrar que a ferramenta desenvolvida cumpriu os objetivos que motivaram a sua construção, servindo como guia para o trabalho do administrador de forma a facilitar a tarefa de manter o banco de dados sempre disponível e com a melhor performance.

Além disto, sua forma de funcionamento através da Web garantiu que o acerto do banco de dados fosse realizado remotamente, mais uma vez comprovando a teoria de que é possível administrar bancos de dados utilizando a Web como meio de comunicação.

8 Conclusões

Durante o desenvolvimento deste estudo, foi possível perceber que a tarefa de administrar bancos de dados exige dedicação, conhecimento e agilidade para resolver possíveis problemas que venham a ocorrer com o banco de dados.

Tanta exigência faz com que os profissionais desta área necessitem trabalhar com ferramentas que os auxiliem a resolver os problemas. No entanto, a evolução das empresas e a sua distribuição por lugares estratégicos para o negócio, faz com que a maioria das ferramentas tradicionais deixem de ser úteis.

Neste trabalho foi projetada, desenvolvida e implantada uma ferramenta que se propõe a auxiliar os administradores de bancos de dados a gerenciar os BD sob sua responsabilidade, independente da localização física de um ou outro, ou seja, que os administradores possam acessar os bancos de dados remotos através da Web.

A implantação da ferramenta comprovou que a idéia de administrar bancos de dados utilizando a Web como meio de comunicação é possível e, em muitos casos, é a única forma de se acessar bancos de dados que estão distantes do administrador.

8.1 Trabalhos Futuros

Muitos consideram a informática uma ciência exata, mas a construção de um aplicativo está muito longe disto. Durante todas as fases do desenvolvimento, sempre ocorrem situações não previstas e a conclusão no tempo previsto passa a ser um desafio.

O desenvolvimento da ferramenta descrita neste trabalho não foi diferente e, portanto, muitos são os aspectos que poderiam ter sido acrescentados e outros, sem dúvida, melhorados.

Através da implantação da ferramenta durante a elaboração do estudo de caso desenvolvido neste trabalho, foi possível perceber as deficiências e, desta forma, motivar os estudos para futuras implementações.

Dentre as futuras implementações e melhorias, para bancos de dados Oracle, podem ser citadas:

- a inclusão de um módulo para efetuar *backup* lógico do banco de dados;
- no módulo de avaliação de desempenho, na opção “Encadeamento de Linhas Tabelas”, criar uma opção para fazer a reorganização das linhas encadeadas das tabelas automaticamente;
- no módulo de avaliação de desempenho, na opção “Definição de Usuários”, inclusão de um botão que faça a alteração dos usuários que estão mal definidos;
- no módulo de gerenciamento de espaço, na opção “Tabelas sem Espaço”, incluir um mecanismo que permita aumentar o espaço das tabelas;
- permitir que o módulo “Área de SQL” execute programas SQL .

Além das alterações de operacionalidade de curto prazo, descritas acima, pode ser considerado um plano de médio prazo, a inclusão de manipulação de outros tipos de objetos dentro dos módulos de gerenciamento de *schema* para bancos de dados Oracle e gerenciamento de *database* para SQL *Server*.

A longo prazo, os objetivos estipulados são:

- estudar mais detalhadamente o SQL *Server* para incluir módulos de gerenciamento específicos para este BD, semelhante ao que já foi implementado para Oracle;
- criação de um repositório de dados para a ferramenta, de forma que seja possível armazenar as estatísticas geradas e assim manter um histórico dos estados dos bancos de dados analisados.

Idéias e determinação não faltam, portanto os benefícios gerados por este trabalho apenas se iniciam com a primeira versão da ferramenta.

Anexo 1 Exemplo do arquivo *server.xml*

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Server>
<xmlmapper:debug level="0" />

<Logger name="tc_log"
            path="logs/tomcat.log"
            customOutput="yes" />
<Logger name="servlet_log"
            path="logs/servlet.log"
            customOutput="yes" />
<Logger name="JASPER_LOG"
             path="logs/jasper.log"
             verbosityLevel = "INFORMATION" />

<ContextManager debug="0" workDir="work" >

       <ContextInterceptor className="org.apache.tomcat.context.AutoSetup" />
       <ContextInterceptor className="org.apache.tomcat.context.DefaultCMSetter"
       />
       <ContextInterceptor
       className="org.apache.tomcat.context.WorkDirInterceptor" />
       <ContextInterceptor className="org.apache.tomcat.context.WebXmlReader" />
       <ContextInterceptor
       className="org.apache.tomcat.context.LoadOnStartupInterceptor" />

       <RequestInterceptor className="org.apache.tomcat.request.SimpleMapper"
       debug="0" />
      <RequestInterceptor
      className="org.apache.tomcat.request.SessionInterceptor" />
      <RequestInterceptor className="org.apache.tomcat.request.SecurityCheck" />
      <RequestInterceptor className="org.apache.tomcat.request.FixHeaders" />

       <Connector className="org.apache.tomcat.service.SimpleTcpConnector">
              <Parameter name="handler"
              value="org.apache.tomcat.service.http.HttpConnectionHandler"/>
              <Parameter name="port" value="8080"/>
      </Connector>

      <Connector className="org.apache.tomcat.service.SimpleTcpConnector">
              <Parameter name="handler"
              value="org.apache.tomcat.service.connector.Ajp12ConnectionHandler"/>
              <Parameter name="port" value="8007"/>
      </Connector>

       <Context     path="/lista"
                    docBase="webapps/lista"
                    crossContext="true"
                    debug="9"
                    reloadable="true" >
      </Context>

       <Context     path="/examples"
                    docBase="webapps/examples"
                    debug="0"
                    reloadable="true" >
      </Context>

</ContextManager>
</Server>
```

Anexo 2 Exemplo do arquivo *web.xml*

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
    "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>

    <servlet>
        <servlet-name>
            ladm
        </servlet-name>
        <servlet-class>
            ListaAdminServlet
        </servlet-class>
    </servlet>

      <servlet>
        <servlet-name>
            lst
        </servlet-name>
        <servlet-class>
            ListaServlet
        </servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>
            ladm
        </servlet-name>
        <url-pattern>
            /admin
        </url-pattern>
    </servlet-mapping>

      <servlet-mapping>
        <servlet-name>
            lst
        </servlet-name>
        <url-pattern>
            /
        </url-pattern>
    </servlet-mapping>


</web-app>
```

Bibliografia

[APA 2001] APACHE SOFTWARE FOUNDATION. **Tomcat-Apache HowTo**. Disponível em: <http://jakarta.apache.org/tomcat/tomcat-3.2-doc/tomcat-apache-howto.html >. Acesso em: 27 out. 2001.

[APC 2001] APACHE SOFTWARE FOUNDATION. **Tomcat – A Minimalistic User's Guide**. Disponível em: <http:// jakarta.apache.org/tomcat/tomcat-3.2-doc/uguide/tomcat_ug.html>. Acesso em: 27 out. 2001.

[APE 2001] APACHE SOFTWARE FOUNDATION. **Release Notes for TOMCAT Version 3.2.4**. [S.l.], 2001.

[APH 2001] APACHE SOFTWARE FOUNDATION. **Developing Applications With Tomcat**. Disponível em: <http://jakarta.apache.org/tomcat/tomcat-3.2-doc/appdev/index.html>. Acesso em: 27 out. 2001.

[BOE 97] BOESING, I. D. **Uma Ferramenta para Tunning e Monitoramento de Banco de Dados Oracle**. 1997. Projeto de Diplomação (Bacharelado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[BRA 99] BRAZ, L. P. **Oracle 7.3 – DBA**. Porto Alegre: Ministério do Exército, 1º Centro de Telemática de Área, 1999.

[CES 96] CESTA, A. A. **Tutorial:** A Linguagem de Programação JAVA. Orientação a Objetos. Disponível em: <http://www.dcc.unicamp.br~aacesta>. Acesso em: 15 fev. 2001.

[CHA 99] CHAN, M. C.; GRIFFITH, S. W.; IASI, A. F. **Java – 1001 Dicas de Programação**. São Paulo: Makron Books, 1999. 714p.

[DEI 2001] DEITEL, H. M.; DEITEL, P. J. **Java, como Programar**. 3.ed. Porto Alegre: Bookman, 2001. 1201p.

[FAQ 2001] FAQ0076. **O que é JDBC?**. Disponível em: <http://nedprof.usf.com.br/~peter/javafaq/jfaq0076.html>. Acesso em: 16 out. 2001.

[FIE 2000] FIELDS, D. K.; KOLB, M. A. **Desenvolvendo na Web com JavaServer Pages**. Rio de Janeiro: Ciência Moderna, 2000. 559p.

[GOO 2001] GOODWILL, J. **Using Tomcat**. Java Web Applications. Disponível em: <http://www.onjava.com/pub/a/onjava/2001/03/15/tomcat.html>. Acesso em: 27 out. 2001.

[HÜB 2001] HÜBNER, J. F. **Acesso a Banco de Dados em Java (JDBC)**. Disponível em: <http://www.lti.pcs.usp.br/~jomi/java/apresentacoes/jdbc/jdbc.ppt>. Acesso em: 16 out. 2001.

[INF 98] INFORMATIVO TÉCNICO NRO 48. **JDBC - Java Database Connectivity**. Disponível em: <http://www.revista.unicamp.br/infotec/informacao/inf48.htm>. Acesso em: 16 out. 2001.

[INT 2001] INTERSIMONE, D. **Apache Tomcat Servlet and JavaServer Pages Development with JBuilder Foundation**. Disponível em: <http://community.borland.com/article/0,1410,22057,00.html>. Acesso em: 27 out. 2001.

[MAI 2002] MANICKAM, M. **A Database Administration Tool with Oracle WebDB.** Disponível em: <http://www.hencie.com/static/whitepapers/webdb_dba_toolkit.pdf>.Acesso em: 26 jun. 2002.

[MAN 2002] MANICKAM, M. **Successful Remote Database Administration? Can it Be a Reality?** Disponível em: <http://www.hencie.com/static/whitepapers/292.pdf >. Acesso em: 26 jun. 2002.

[MAM 2002] MANICKAM, M. **Oracle Application iDBA Toolset – Using Designer.** Disponível em: <http://www.hencie.com/static/whitepapers/Oracle_apps_idba_toolkit.pdf>. Acesso em: 26 jun. 2002.

[NEW 97] NEWMAN, A. **Usando Java – O Guia de Referência mais Completo**. Rio de Janeiro: Campus, 1997. 861p.

[OLI 2001] OLIVEIRA, D. V. de. **JDBC - Java Database Connectivity.** Videira: Universidade do Oeste de Santa Catarina, 2001. p.1-6.

[ORA 2000] ORACLE CORPORATION. **Oracle Enterprise Manager**. Oracle Standard Management Pack Installation, Release 2.1 for Windows NT, Windows 95, and Windows 98. Redwood City, 2000. 22p.

[ORC 2000] ORACLE CORPORATION. ***i*SQL*Plus Installation and User´s Guide**. Release 8.1.7 Beta for Windows. Redwood City, 2000.

[ORL 2001] ORACLE CORPORATION. **Oracle Enterprise Manager**. How Enterprise Manager Works. Redwood City, 2000.

[ORE 99] ORACLE CORPORATION. **Oracle 8*i***. JDBC Developer´s Guide and Reference – Release 2 (8.1.6). Redwood City, 1999.

[ORM 2000] ORACLE CORPORATION. **Meeting the IT Workforce Shortage:** Oracle *i*DBA Support Service. Redwood City, 2000.

[ORO 2000] ORACLE CORPORATION. **Oracle 8:** Database Administration. Redwood City, 2000. v. 1-3.

[ORR 99] ORACLE CORPORATION. **Oracle8™ Server Concepts – Release 8.0**. Redwood City, 1999.

[PED 2001] PEDRON, S. L.; GREMELMEIER, C. H. **Utilização da linguagem de programação Java com bancos de dados relacionais**. Disponível em: <http://www.inf.uri.com.br/utilizacao.htm>. Acesso em: 16 out. 2001.

[RAM 2001] RAMON, F. **JDBC 2 – Acesso a bancos de dados usando a linguagem Java**. Guia de Consulta Rápida. Rio de Janeiro: Novatec, 2001. 96p.

[SAR 2000] SARIN, S. **Oracle DBA:** dicas e técnicas. Rio de Janeiro: Campus, 2000. 706p.

[SCH 2001] SCHERER, A. P. Z. **Um estudo sobre a administração de banco de dados**. 2001. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[SIL 99] SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados**. São Paulo: Makron Books, 1999. 778p.

[STO 97] STOUT, R. **Dominando a World Wide Web**. São Paulo: Makron Books, 1997. 710p.

[WML 2001] WMLCLUB – Wireless Markup Language Club. **Instalação e Configuração do Tomcat**. Disponível em: <http://www.wmlclub.com.br/articulos/instalatomcat.htm>. Acesso em: 27 out. 2001.