FUNDAÇÃO DE ENSINO “EURÍPIDES SOARES DA ROCHA”
CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA - UNIVEM
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RODRIGO KAMADA PADOVEZI DIAS

PROGRAMA DE APLICAÇÃO PARA MANIPULAÇÃO DE DADOS EM UM SGBDR USANDO JDBC - JAVA

Marília
2005

RODRIGO KAMADA PADOVEZI DIAS

PROGRAMA DE APLICAÇÃO PARA MANIPULAÇÃO DE DADOS EM UM SGBDR USANDO JDBC - JAVA

Monografia final de conclusão de curso apresentada ao Curso de Ciência da Computação, mantido pela Fundação de Ensino "Eurípides Soares da Rocha", para obtenção do título de Bacharel em Ciência da Computação.

Orientador:
Prof. Dr. Edmundo Sérgio Spoto

Marília
2005

DIAS, Rodrigo Kamada Padovezi

Programa de aplicação para manipulação de dados em um SGBDR usando JDBC – JAVA / Rodrigo Kamada Padovezi Dias; orientador: Edmundo Sérgio Spoto. Marília, SP:[s.n.], 2003

51 f.

Monografia – Centro Universitário Eurípedes de Marília – Fundação de Ensino Eurípedes Soares da Rocha.

1.Java 2.JSP 3.JDBC 4.Oracle9i

Cdd: 005.1151

RODRIGO KAMADA PADOVEZI DIAS

PROGRAMA DE APLICAÇÃO PARA MANIPULAÇÃO DE DADOS EM UM SGBDR USANDO JDBC - JAVA

Banca examinadora da monografia apresentada ao Curso de Ciência da Computação UNIVEM/F.E.E.S.R., para obtenção do título de Bacharel em Ciência da Computação.

Resultado: _____

ORIENTADOR: Prof. Dr. Edmundo Sérgio Spoto

1º EXAMINADOR: Profa. Dra. Ana Paula Piovesan Melchiori Peruzza

2º EXAMINADOR: Profa. Dra. Kalinka Regina Lucas Jaquie Castelo Branco

Marília, 29 de Novembro de 2005.

AGRADECIMENTOS

Agradeço a DEUS, o meu guia para toda a vida, pela força de superar desafios.

À minha família, em especial aos meus pais Paulo Sérgio Padovezi Dias e Rose Mary Kamada Padovezi Dias pela compreensão, ajuda e educação.

À minha irmã Patrícia Kamada Padovezi Dias que muitas vezes me incentivou.

À minha namorada Raquel Roso Dezan pela presença, incentivo e força na realização deste.

À minha tia Inove Kamada pela paciência que demonstrou ter dedicando parte de seu tempo no esclarecimento de algumas das dúvidas que surgiram ao longo do desenvolvimento deste trabalho.

Amo todos vocês.

Ao Prof. Dr. Edmundo Sérgio Spoto, meu orientador, pela estrutura, conhecimento e companheirismo que me proporcionou para o desenvolvimento desta.

A todas as pessoas que me ajudaram direta e indiretamente em mais este trabalho.

DIAS, Rodrigo Kamada Padovezi. **Programa de aplicação para manipulação de dados em um SGBDR usando JDBC – JAVA**. 2005, 51 f. Monografia – Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípedes Soares da Rocha, Marília, 2005.

RESUMO

Este trabalho apresenta um estudo sobre as tecnologias Java e sistemas gerenciadores de banco de dados Oracle 9i para o desenvolvimento de uma aplicação para manipular base de dados Oracle 9i usando JDBC. Com o avanço da Internet, houve uma demanda no desenvolvimento de aplicações para a Internet, e para atender as necessidades surgiram novas tecnologias para a construção de páginas dinâmicas como a Java Server Pages – JSP, fornecendo toda infra-estrutura necessária para a resolução de problemas de distribuição de tarefas executadas. Inserido dentro deste contexto, o presente trabalho tem com propósito abordar as tecnologias da plataforma Java e banco de dados Oracle 9i, voltadas ao desenvolvimento de uma aplicação para Internet, abstraindo detalhes de um usuário leigo na manipulação de sistemas gerenciadores de banco de dados, porém sem deixar de privilegiar a coerência e apresentar a aplicação de conceitos importantes.

Palavras-chave: Java, JSP, JDBC, Oracle9i.

DIAS, Rodrigo Kamada Padovezi. **Program of application for manipulation of data in a SGBDR using JDBC – JAVA**. 2005, 51 f. Monografia – Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípedes Soares da Rocha, Marília, 2005.

ABSTRACT

This work presents a study about the Java technologies and database management systems Oracle 9i for the development of an application to manipulate database Oracle 9i using JDBC. With the advance of the Internet, it had a demand in the development of applications for the Internet, and to take care of the necessities had appeared new technologies for the construction of dynamic pages as the Java Server Pages - JSP, supplying all necessary infrastructures the resolution of problems of distribution of executed tasks. Inserted inside of this context, the present work has with intention to approach the technologies of the Java platform and database Oracle 9i, come back to the development of an application toward Internet, abstracting details of a lay user in the manipulation of database management systems, however leaving to privilege the coherence and to present the application of important concepts.

Keywords: JAVA, JSP, JDBC, Oracle9i.

DIAS, Rodrigo Kamada Padovezi. **Programa de aplicación para manipulación de dados en uno SGBDR usando JDBC – JAVA**. 2005, 51 f. Monografia – Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípedes Soares da Rocha, Marília, 2005.

RESUMEN

Este trábalo presenta un estudio de las tecnologías Java y sistemas gerenciales de banco de datos Oracle 9i para el desarrollo de una aplicación para manipular la base de datos Oracle 9i usando JDBC. Con el avanzo de la Internet, huvo una demanda en el desarrollo de aplicaciones para Internet y para atender las necesidades surgieron nuevas tecnologías para la construcción de paginas dinámicas como Java Server Pages – JSP, forneciendo toda infra-estructura necesaria para la resolución de problemas de distribución de tareas ejecutadas. Insiriendo dentro de este contexto, el referido trabajo que tiene como propósito abordar las tecnologías de plataforma Java y base de datos Oracle 9i, visando el desarrollo de una aplicación para Internet, abstrayendo detalles de un usuario lego en la manipulación de sistemas gerenciadores de banco de datos, sin dejar de privilegiar la coherencia y demostrar la aplicación de conceptos importantes.

Palabras-Clave: JAVA, JSP, JDBC, Oracle9i.

GLOSSÁRIO

API	- Aplication Programming Interface
ASF	- Apache Software Foundation
CSS	- Cascading Style Sheets
DDL	- Data Definition Language
DML	- Data Manipulation Language
GIF	- Graphics Interchange Format
HTML	- HyperText Markup Language
HTTP	- HyperText Transfer Protocol
J2SE	- Java 2 Standard Edition
JAR	- Java Archive
JDBC	- Java Database Connectivity
JRE	- Java Runtime Environment
JS	- JavaScript
JSP	- Java Server Pages
JVM	- Java Virtual Machine
SQL	- Structured Query Language
URI	- Uniform Resource Identifier
URL	- Uniform Resource Locator
XML	- Extensible Markup Language
WWW	- World Wide Web

Sumário

GLOSSÁRIO 9
1. INTRODUÇÃO 12
1.1 Contexto e motivação 12
1.2. Objetivo Geral 13
1.3. Objetivos Específicos 13
1.4. Organização 14
2. DEFINIÇÃO E TERMINOLOGIA 15
2.1 Plataforma Java 15
2.2 Arquitetura Java 16
2.3 Java Server Pages 17
2.4 Arquitetura do Banco de Dados Oracle 18
2.5 API JDBC 19
2.6 Tomcat 26
2.7 Considerações Finais 31
3. FERRAMENTA jspOraAdmin 32
3.1 Arquitetura 32
3.2 Funcionamento da Ferramenta 33
4. CONCLUSÃO 48
5. REFERÊNCIA BIBLIOGRÁFICAS 49

Índice de Figuras

Figura 1 - Plataforma Java [HÓLM, 2002] 16
Figura 2 - Execução de uma Java Server Pages 18
Figura 3 - Arquitetura do banco de dados Oracle 19
Figura 4 - Interfaces e classes presentes no pacote java.sql [HÓLM, 2002] 21
Figura 5 - Comunicação com o banco de dados [HÓLM, 2002] 22
Figura 6 - Arquitetura dos drivers JDBC [HÓLM, 2002] 23
Figura 7 - Arquitetura Oracle Thin Driver [HÓLM, 2002] 25
Figura 8 - Código fonte para a conexão com o banco de dados [HÓLM, 2002] 26
Figura 9 - Estrutura de diretórios do Tomcat 29
Figura 10 - Start do Tomcat 30
Figura 11 - Configuração da porta 30
Figura 12 - Configuração da pasta da aplicação 31
Figura 13 - Arquitetura da Ferramenta jspOraAdmin 32
Figura 14 - Tela de login da ferramenta jspOraAdmin 34
Figura 15 - Tela de abertura e criação da nova tabela 35
Figura 16 - Definição dos dados para criação da tabela 36
Figura 17 - Estrutura das tabelas do banco de dados 37
Figura 18 - Execução de comandos SQL no banco de dados 38
Figura 19 - Exportação dos dados contidos no banco de dados 39
Figura 20 - Dados exportados do banco de dados 40
Figura 21 - Procura registros no banco de dados 41
Figura 22 - Registros encontrados no banco de dados 42
Figura 23- Estrutura das colunas da tabela 43
Figura 24 - Confirmação para a criação de uma chave primária 43
Figura 25 - Criação de novos campos na tabela 44
Figura 26 - Visualização dos registros da tabela 45
Figura 27 - Execução de comandos SQL na tabela 45
Figura 28 - Procura registros na tabela 46
Figura 29 - Insere registros na tabela 47

1. INTRODUÇÃO

Este trabalho apresenta uma aplicação que ajudará no processo de manipulação de um banco de dados Oracle 9i.

Quando na manipulação de um banco de dados, eventuais dúvidas e falta de conhecimento que surjam podem se apresentar como um empecilho muito grande, dificultando ou, até mesmo, impossibilitando a execução de comando SQL, neste sentido, o ***jspOraAdmin*** tem papel fundamental, ao viabilizar ao usuário que está manipulando o banco de dados a superação de tais obstáculos, e, conseqüentemente, a execução de comandos SQL no banco de dados, podendo ser comandos DML ou DDL.

Neste capítulo é apresentado o contexto no qual o trabalho está inserido, assim como a motivação para realizá-lo e os objetivos a serem atingidos.

1.1 Contexto e motivação

O surgimento de novas tecnologias e de novas formas de fazer tecnologia tem passado por grandes mudanças. De maneira geral, essas mudanças representam quebra de paradigmas e alteram as estruturas do desenvolvimento mundial.

Nos últimos anos, diferentes tipos de tecnologias para o desenvolvimento de aplicações, assim como linguagens de programação e de banco de dados, vêm sendo disputados por grandes empresas, dos quais destacamos sem dúvida, a plataforma Java para o desenvolvimento de programas e o sistema de gerenciamento de banco de dados Oracle 9i para a manipulação de dados.

Neste cenário, surgiu a proposta para o desenvolvimento de uma aplicação que permitisse unir essas grandes tecnologias permitindo a interação com a Internet e auxiliando a manipulação de uma base de dados de forma prática.

A utilização da aplicação associada ao desenvolvimento das tecnologias mais recentes é significativa para alcançar o objetivo almejado deste projeto.

Este foi o ponto de partida para o desenvolvimento do ***jspOraAdmin***, incluso na ferramenta deste trabalho.

1.2. Objetivo Geral

Possibilitar ao usuário uma visão sobre a manipulação de dados em um sistema gerenciador de banco de dados relacional, a partir de um conjunto de regras pré-estabelecidas pelo banco de dados Oracle 9i, utilizando conexão JDBC.

O usuário desta ferramenta terá condições de manipular dados desejados, incluindo toda a parte funcional de manipulação de tabelas e registros, sem a necessidade de conhecer comandos SQL.

1.3. Objetivos Específicos

- Proporcionar ao usuário manipular o banco de dados que estará conectado, executando comandos SQL ou simplesmente selecionar as opções mostradas na tela.
- Auxiliar na criação, alteração ou exclusão de tabelas, mostrando as funcionalidades necessárias para a manipulação do mesmo.

- Ele poderá acessar uma ou mais tabelas, inserir, eliminar ou atualizar dados, etc.
- Os comandos executados serão realizados e retornados em tempo de execução para que o usuário entenda e verifique o resultado exibido na tela.

1.4. Organização

Esta monografia está dividida em seis capítulos. Neste primeiro capítulo é introduzido o assunto tema da pesquisa e também são abordados: o contexto, a motivação e o objetivo geral e o objetivo específico, da mesma.

A seguir, o segundo capítulo apresentam, de forma resumida, a fundamentação teórica necessária para a elaboração do trabalho, com uma abordagem na plataforma Java, características do Oracle, e funcionamento do Tomcat.

No terceiro capítulo, é apresentado o projeto da ferramenta de aplicação jspOraAdmin mostrando todas as técnicas utilizadas para a manipulação de tabelas e registros.

O quarto capítulo apresenta o estudo de caso com os dados de alguns dicionários de dados utilizados no desenvolvimento da ferramenta e informações das principais ferramentas da Oracle.

No quinto capítulo, são apresentadas as conclusões que este trabalho permitiu alcançar e sugestões para futuros trabalhos.

E por último, no sexto capítulo, o material consultado para a realização da mesma.

2. DEFINIÇÃO E TERMINOLOGIA

2.1 Plataforma Java

Java é uma linguagem orientada a objetos, portável, concorrente e distribuída que se tornou popular, inicialmente, por permitir a construção de aplicações para Internet.

O bytecode gerado pelo compilador para a sua aplicação específica pode ser transportado entre plataformas distintas que suportam Java. Não é necessário recompilar um programa para que ele rode em uma máquina ou em um sistema diferente. Esta portabilidade é importante para a criação de aplicações para a heterogênea Internet.

A presença de uma coleta automática de lixo evita erros comuns que os programadores cometem quando são obrigados a gerenciar diretamente a memória. A eliminação do uso de ponteiros, em favor do uso de vetores, objetos e outras estruturas traz benefícios em termos de segurança. O programador é proibido de obter acesso à memória que não pertence ao seu programa, além de não ter chances de cometer erros comuns tais como "*reference aliasing*" e o uso indevido de aritmética de ponteiros.

Java é muito mais eficiente que grande parte das linguagens, e com a evolução da linguagem estão sendo criados geradores de "*bytecode*" cada vez mais otimizados que trarão as marcas de performance da linguagem mais próximas dessas linguagens.

A seguir, discutiremos alguns dos principais aspectos da linguagem Java, assim como sua arquitetura.

2.2 Arquitetura Java

Os programas Java (arquivos .java) são compilados por compiladores Java gerando arquivo de classes Java (arquivos .class). Os arquivos de classe são os executáveis do Java. Ao executar um programa Java, as classes produzidas são carregadas e executadas dentro de um JRE (*Java Runtime Environmment*), ou seja, ambiente de tempo de execução Java. [HÓLM, 2002]

Esse JRE compõe a plataforma Java e consiste em duas partes. A primeira é a JVM (*Java Virtual Machine*) e a segunda é a API (*Application Programmer Interface*) Java, ou seja, interface de programas aplicativos. A Figura 1 ilustra isso.

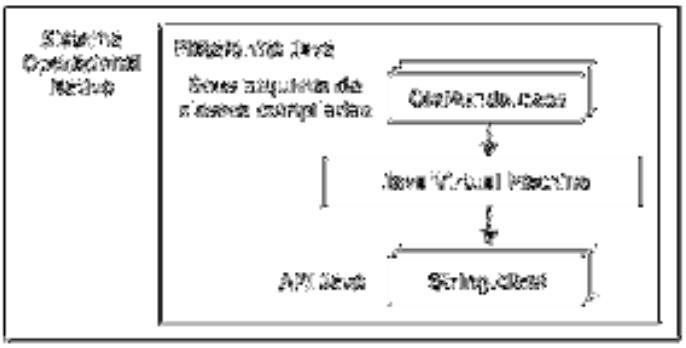

Figura 1 - Plataforma Java [HÓLM, 2002]

A JVM é basicamente uma camada que está entre os programadores e os sistemas operacionais. Os detalhes dependentes da plataforma são tratados pela camada JVM.

As diferenças entre as diversas plataformas estão na API Java que é incluída. Por exemplo, a edição *Standard* contém APIs para a criação de formas gráficas (*Swing*) mas não contém APIs *Enterprise* tais como as APIs *Servlet* e *Enterprise JavaBeans*. Todas as plataformas Java 2 devem conter as APIs básicas. Classes essenciais como String e Object não são opcionais e, portanto, estão presentes nas APIs básicas.

2.3 Java Server Pages

JSP é uma tecnologia Java para desenvolvimento de aplicações Web semelhante ao *Microsoft Active Server Pages* (ASP), porém tem a vantagem da portabilidade de plataforma que o Java provê, o que lhe permite ser executada em outros Sistemas Operacionais.

O propósito da JSP é prover um método declarativo e centrado em apresentação para o desenvolvimento de *Servlets*. Programadores familiarizados com a tecnologia *Servlet* sabem que não há nada que o JSP ofereça que não possa ser conseguido com o uso dos *Servlets* puros. Porém, a *Java Server Pages* facilitam o desenvolvimento da aplicação basicamente por uma razão: o programador pode separar a programação lógica (parte dinâmica) da programação visual (parte estática). Essa separação, além de simplificar o desenvolvimento em si e tornar o código menos propenso a erros (possivelmente também mais confiável), permite que cada parte seja feita de forma independente. Assim, um *webdesigner* e um programador podem trabalhar em conjunto para um mesmo projeto.

Para construir uma aplicação usando JSP escreve-se o texto HTML e incluem-se códigos JSP entre tags JSP. Essas tags normalmente começam com "<%" e terminam com "%>". O código para formatação de página que é usado nos aplicativos *Servlets* não é mais necessário, o código Java pode ser escrito dentro da própria página HTML. Isso prejudica um pouco a legibilidade do código, mas, em nome da flexibilidade, muitas vezes é recomendado.

Quando uma página JSP é requisitada através de um navegador Web, esta página é executada pelo servidor, que carrega um aplicativo *Servlet* equivalente ao código da página JSP (um arquivo Java que será interpretado pelo compilador) e então uma página HTML é gerada e enviada de volta para o cliente. A execução de uma página JSP se transforma

necessariamente em um *Servlet*, o que mostra que os *Servlets* têm a mesma capacidade de desenvolvimento da JSP, só que escritos de forma um pouco mais complexa.

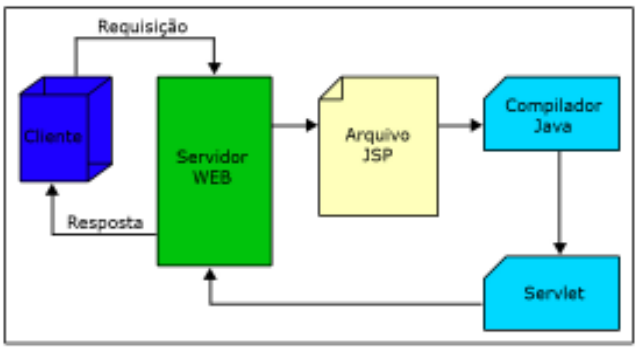

Figura 2 - Execução de uma Java Server Pages

A Figura 2 mostra a requisição de um cliente para o servidor para a execução de uma página JSP. O compilador Java interpretará o arquivo Java e então uma página HTML é gerada e enviada de volta para o cliente.

2.4 Arquitetura do Banco de Dados Oracle

Um banco de dados Oracle é formado por diversos elementos, incluindo estruturas de memória, processos especiais que aceleram as execuções e os mecanismos de recuperação que permitem DBA restaurar os sistemas após a ocorrência de problemas aparentemente irrecuperáveis. [COUCHMAN, 2003]

O servidor de banco de dados Oracle é formado por muitos componentes diferentes, alguns deles são estruturas de memória, enquanto outros são processos de plano de fundo que executam determinadas tarefas nos bastidores. Também há recursos de disco que armazenam dados que os aplicativos utilizam para monitorar os dados de uma organização inteira e recursos especiais desenvolvidos para permitir a recuperação de dados de problemas que vão

desde uma entrada incorreta até uma falha no disco. As estruturas de memória e os processos de plano de fundo constituem uma instância Oracle, enquanto a instância Oracle com as estruturas restantes constitui um banco de dados Oracle.

Na Figura 3 são apresentados a arquitetura do banco de dados Oracle, com os diversos tipos de clientes ou servidores estabelecendo a conexão e gerando a instância SQL para a base de dados Oracle.

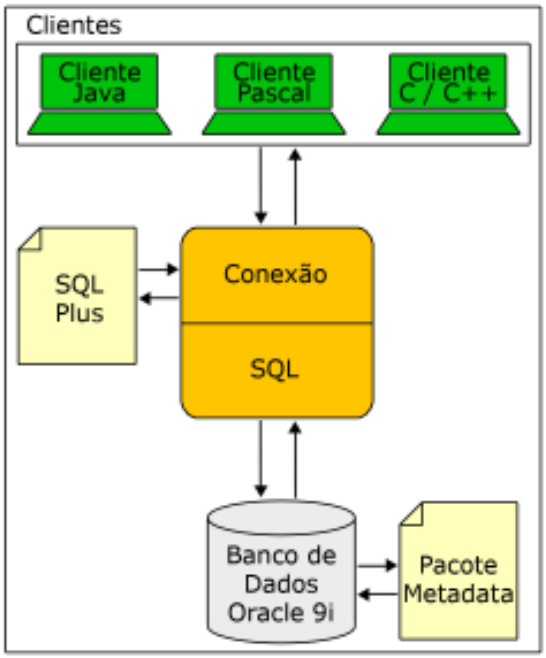

Figura 3 - Arquitetura do banco de dados Oracle

2.5 API JDBC

A biblioteca de classes e as interfaces da JDBC são usadas dentro de um programa Java para abrir conexões com determinados bancos de dados, executarem instruções DML ou DDL, obter resultados e chamar procedimentos e funções armazenados, assim como outras operações normais de banco de dados. [HÓLM, 2002]

A JDBC é uma API baseada em objetos em Java para acesso a banco de dados, e seu objetivo é ser um padrão ao qual desenvolvedor Java e fornecedores de banco de dados possam aderir.

Não há código de implementação nos métodos das classes JDBC. Em vez disso, um *driver* JDBC específico de um fornecedor traduz as chamadas da API em operações para o banco de dados desse fornecedor específico. Na teoria, isso significa que a chamada JDBC "isola" os programadores de Java das características particulares do sistema de banco de dados que está sendo usado, implicando que seria possível portar dentre diversos bancos de dados sem ter que refazer o código JDBC para cada banco de dados específico.

Na verdade, esse foi o princípio básico do projeto JDBC, válido para o Java atingir o objetivo de ser uma plataforma do tipo "escreva uma vez e execute em qualquer lugar", e é claro que esse projeto é muito valioso para o programador. Na prática, fazer código JDBC funcionar bem e de maneira coerente em diversas fontes de dados requer trabalho e desenvolvimento consideráveis, pois todos os bancos de dados têm suas diferenças. [HC02]

Um dos princípios fundamentais do projeto JDBC era tornar prática a construção de *drivers* JDBC com base em outras APIs de banco de dados.

A API básica da JDBC, contendo todas as classes essenciais, interfaces e exceções para acessar um banco de dados, é fornecida no pacote java.sql, que compõe uma parte da biblioteca de classes Java.

No diagrama apresentado na Figura 4 são mostradas algumas das interfaces e classes básicas presentes no pacote java.sql:

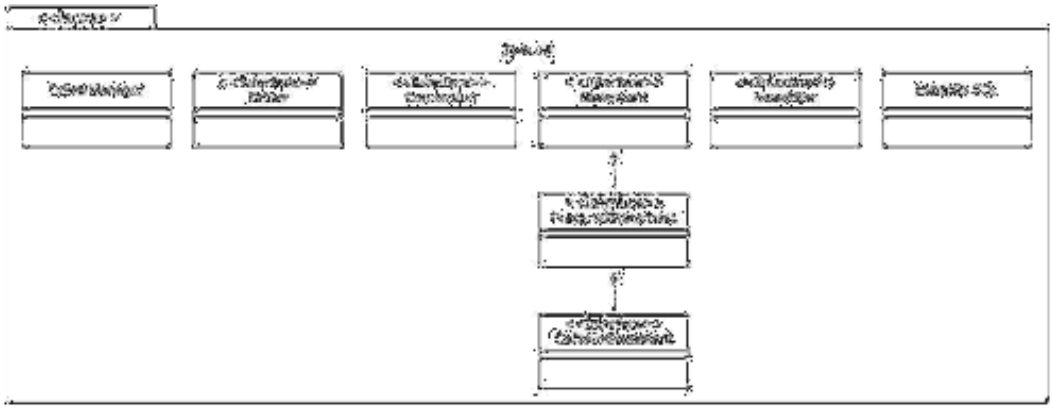

Figura 4 - Interfaces e classes presentes no pacote java.sql [HÓLM, 2002]

- java.sql.DriverManager – esta classe lida com os *drivers* JDBC e é usada para registrar *drivers* e criar conexões JDBC.
- java.sql.Driver – esta interface representa o *driver* JDBC e deve ser implementada por todos os fornecedores de *drivers*. A classe oracle.jdbc.OracleDriver é a classe que implementa a interface *Driver* nos *drivers Oracle* JDBC.
- java.sql.Connection – esta interface representa uma conexão de banco de dados e possui métodos para criar instruções SQL para executar operações SQL regulares. As instruções SQL são sempre executadas no contexto de *Connection*.
- java.sql.Statement – esta interface fornece métodos para a execução de instruções SQL e no contexto de uma conexão com um determinado banco de dados.Há duas subinterfaces importantes:
- java.sql.PreparedStatement – permite a execução de instruções pré-interpretadas, melhorando o desempenho em todas as operações de banco de dados.

- java.sql.CallableStatement – permite a execução de procedimentos armazenados, tais como PL/SQL e procedimentos armazenados Java.
- java.sql.ResultSet – esta interface contém de fornece métodos para acessar linhas encontradas em uma consulta SQL retornada pela execução de comandos (*Statement)*. Dependendo do tipo de *ResultSet* que estiver sendo usado, também poderemos ter métodos para alterar, modificar e manipular os dados obtidos.
- java.sql.SQLException – uma exceção que provê o acesso a todas as informações relacionadas a erros de banco de dados.

Os *drivers* Oracle JDBC fornecem recursos adicionais aos recursos padrão da JDBC, tais como, extensões como tipos de dados específicos do Oracle e suportam melhorias de desempenho através da execução de lotes e da busca prévia de linhas.

O banco de dados normalmente tem protocolos de rede de terceiros para a comunicação entre o cliente e o banco de dados. Cada *driver* JDBC possui o código necessário para conectar e interagir com um determinado banco de dados.

A classe DriverManager, de forma automática para o programador, comunica-se com o banco de dados e realiza a operação de banco de dados requisitada e retorna todos os dados solicitados, conforme é mostrada na Figura 5:

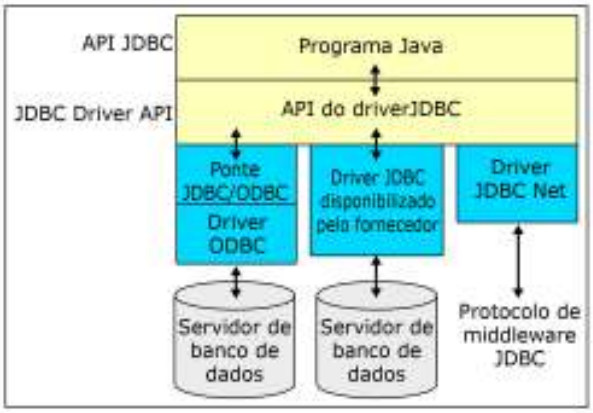

Figura 5 - Comunicação com o banco de dados [HÓLM, 2002]

A Oracle fornece quatro tipos de *drivers* JDBC que podem ser usados em diversas situações. A classe *DriverManager* é usada para abrir novas conexões e as informações necessárias dependem do *driver* de banco de dados que estiver sendo usado.

Os *drivers* JDBC fornecidos atualmente são dois da parte do cliente: JDBC *Thin driver* (Tipo 2) e *driver* JDBC OCI (Tipo 4), e dois da parte servidor: *Server-Side Thin driver* (Tipo 4) e *driver Server-Side Internal* (Tipo 2).

No diagrama apresentado na Figura 6 é mostrada a arquitetura geral dos *drivers* JDBC da Oracle 9i.

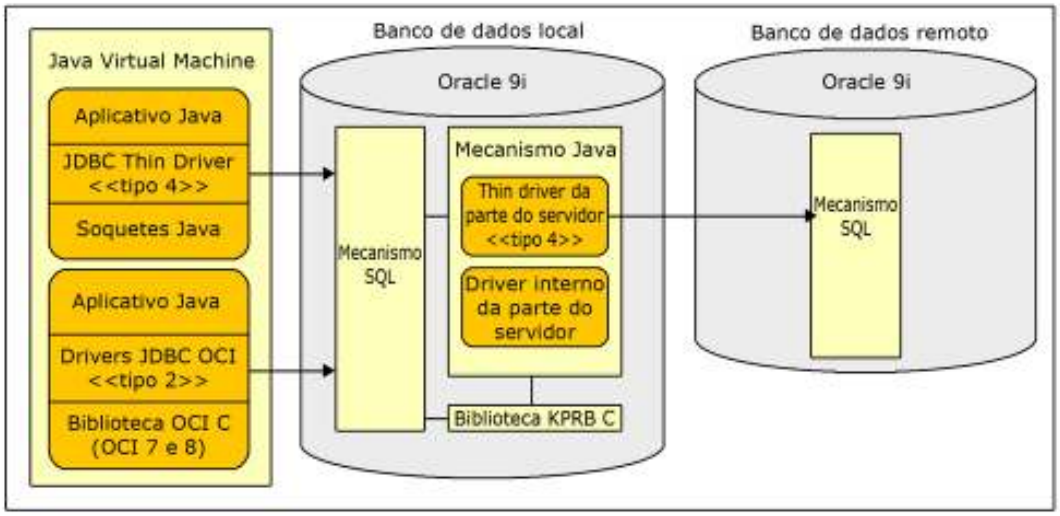

Figura 6 - Arquitetura dos drivers JDBC [HÓLM, 2002]

Todos os *drivers* do Oracle oferecem basicamente os mesmos recursos, suportando o padrão JDBC 1.22 e as extensões Oracle para JDBC 2.0, ou seja, as mesmas APIs e sintaxe.

A diferença real entre os *drivers*, é a maneira pela qual se conectam o banco de dados e como os dados são transferidos.

O *thin driver Oracle* é um *driver* do tipo 4 e, portanto, é portável de plataformas. Utiliza soquetes Java padrão para se comunicar diretamente com o banco de dados e é usado

na parte do cliente (fora do banco de dados). Como não exige qualquer software adicional, o *thin driver* é adequado a aplicativos de cliente, assim como para *applets* Java. [HC02]

Para conectar com o Oracle, é necessário fornecer à conexão JDBC detalhes suficientes para identificar o banco de dados com o qual desejamos conectar.

As informações necessárias sobre o banco de dados Oracle são:

- HOST – O nome ou o endereço IP da máquina onde o banco de dados Oracle está instalado. Se um banco de dados for usado, *localhost* poderá ser usado como nome do servidor.
- PORT – A porta definida para um *listener* de banco de dados válido. A porta padrão é 1521.
- SERVICE_NAME – O nome do serviço (*service name*) é o novo esquema de nomeação implementado para identificar um banco de dados. Trata-se de uma representação lógica de um banco de dados.
- TNSENTRY – O TSENTRY deve ser uma entrada válida no arquivo tnsnames.ora, que se encontra no diretório ORACLE_HOME\network\admin. Ele contém todas as informações necessárias para conectar-se à base de dados que utilizará.
- Nome do usuário e senha do banco de dados.

O URL JDBC varia dependendo do banco de dados e do *driver* JDBC usado, contudo, todos os URLs JDBC assumem a forma:

jdbc:<subprotocol>:<subname>

O esquema jdbc: indica que o URI se refere a uma fonte de dados JDBC. O <*subprotocol*>: identifica qual *driver* JDBC deve ser usado. O <*subname*> é o identificador da fonte de dados, cujo formato será determinado pelo *driver* JDBC.

O formato básico de um URL para o *Oracle thin driver* é:

jdbc:oracle:thin:@HOST_NAME:PORT:SERVICE_NAME

O valor do <subprotocol> é oracle:thin:, indicando o uso do *thin driver* JDBC. Neste projeto, o URL utilizado é o seguinte:

jdbc:oracle:thin:@localhost:1521:oracle9i

no qual, conforme descrito acima, o *localhost* é o nome da máquina onde está hospedado o banco de dados Oracle, 1521 é a porta de escuta do banco de dados e oracle9i identifica o nome do banco de dados específico a qual desejamos conectar, conforme mostrado na Figura 7.

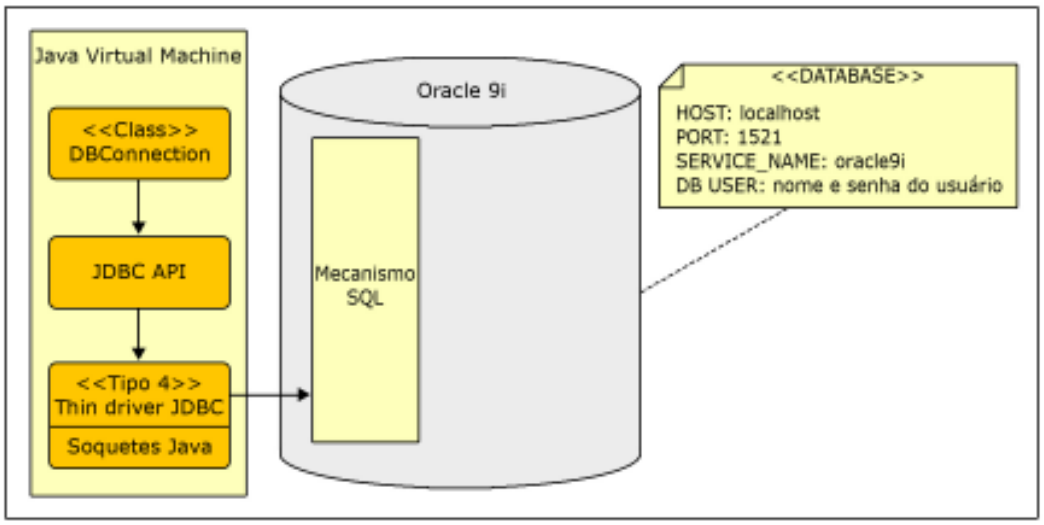

Figura 7 - Arquitetura Oracle Thin Driver [HÓLM, 2002]

Na Figura 8 é mostrado o código fonte de como é feita a conexão de acordo com os dados informados na tela de *login*. Esses dados como, usuário, senha e nome do banco de dados, porta e endereço no qual estabelecerá a conexão, são carregados por meio de uma sessão. Ao definir o *driver* e montar a URL, se os dados informados estirem corretos, é estabelecida a conexão com o banco de dados Oracle 9i.

```
<%@ page import="java.sql.*" %>

<%
HttpSession sessao = request.getSession();

String erro           = null;
String driver         = "oracle.jdbc.driver.OracleDriver";
String sessao_usuario = sessao.getAttribute("sessao_usuario").toString();
String sessao_senha   = sessao.getAttribute("sessao_senha").toString();
String sessao_bd      = sessao.getAttribute("sessao_bd").toString();
String sessao_host    = sessao.getAttribute("sessao_host").toString();
String sessao_porta   = sessao.getAttribute("sessao_porta").toString();
String url            = "jdbc:oracle:thin:@" + sessao_host + ":" + sessao_porta + ":" + sessao_bd;
Connection conexao    = null;
ResultSet resultado   = null;
Statement declaracao  = null;
String sql = null;

try {
        Class.forName(driver).newInstance();
        conexao = DriverManager.getConnection(url,sessao_usuario,sessao_senha);
}
catch (Exception e) {
        response.sendRedirect("index.jsp");
}
%>
```

Figura 8 - Código fonte para a conexão com o banco de dados [HÓLM, 2002]

No próximo item deste capítulo será apresentado o servidor de páginas utilizado para o desenvolvimento da ferramenta.

2.6 Tomcat

O Tomcat, desenvolvido pela Fundação Apache Software (ASF), atualmente está na Versão 5.5.12, permite a execução de aplicações voltadas para WEB com suporte a CGI. Sua principal característica técnica é estar centrada na linguagem de programação Java, mais especificamente nas tecnologias de Servlet e de Java Server Pages. Em sua utilização, seja para fins comerciais ou não, é livre.

Foi escolhido por ser um servidor rápido, muito utilizado por várias empresas de tecnologia de desenvolvimento WEB e também fácil de ser configurado e utilizado.

Por estar escrito em Java, o Tomcat necessita que a versão Java 2 Standard Edition (J2SE) esteja instalada no mesmo computador onde será executado. No entanto, não basta ter

a versão *runtime* de Java instalada, pois o Tomcat necessita compilar (e não apenas executar) programas escritos em Java. O Projeto Jakarta da ASF, do qual o subprojeto Tomcat é o representante mais ilustre, tem como objetivo o desenvolvimento de soluções código aberto baseadas na plataforma Java.

O desenvolvimento de uma aplicação WEB a ser executada pelo Tomcat implica no domínio das seguintes linguagens:

- Java: Todos os algoritmos da aplicação devem ser desenvolvidos na linguagem Java.
- HTML: A interface da aplicação, isto é, aquilo que o usuário visualizará e/ou manipulará em sua tela, é construída na forma de páginas escritas em HTML e visualizadas através do navegador do usuário. Esta tarefa normalmente é delegada ao WEB designer.
- XML: Qualquer aspecto relacionado à configuração da aplicação deve ser expresso por meio da linguagem XML em um arquivo chamado web.xml. Os dados de configuração podem ser usados tanto pelo Tomcat como pela aplicação.

O servidor Tomcat tem a habilidade de converter automaticamente qualquer página JSP em um *servlet* equivalente. Em outras palavras, o Tomcat é capaz de criar código fonte Java a partir de um documento HTML.

Do ponto de vista técnico, Tomcat é a implementação referência das especificações das tecnologias de servlets e JSP criadas pela Sun. A Versão 5.5.12 do Tomcat implementa as especificações *Servlet* 2.4 e JSP 2.0.

Do ponto de vista operacional, a principal finalidade das tecnologias de *servlet* e JSP é permitir a criação dinâmica de conteúdos. A dinâmica, em um cenário típico, funciona do seguinte modo:

- Um usuário, no seu navegador, solicita algum documento (indicado por um URL) a um servidor Tomcat.
- O servidor, ao receber uma solicitação (URL) do usuário, executa o *servlet* ou JSP correspondente àquela URL (a associação entre URL e *servlet* ou JSP é especificada no arquivo web.xml). O conteúdo gerado pelo *servlet* ou JSP, normalmente um documento no formato HTML, é uma combinação de tags HTML (incluídos explicitamente) e o resultado de algum processamento (por exemplo, algoritmo Java e/ou acesso a um banco de dados).
- O usuário recebe o conteúdo gerado pelo servidor Tomcat e o exibe através do seu navegador.

Em termos práticos, o Tomcat pode ser usado isoladamente, assumindo o papel de um servidor WEB, ou em conjunto com outro servidor (como o Apache). Neste caso, o Apache atende a requisições de páginas estáticas enquanto que o Tomcat atende a requisições de páginas dinâmicas.

Assim como todo servidor WEB existe uma estrutura de diretórios que é representada conforme a Figura 9.

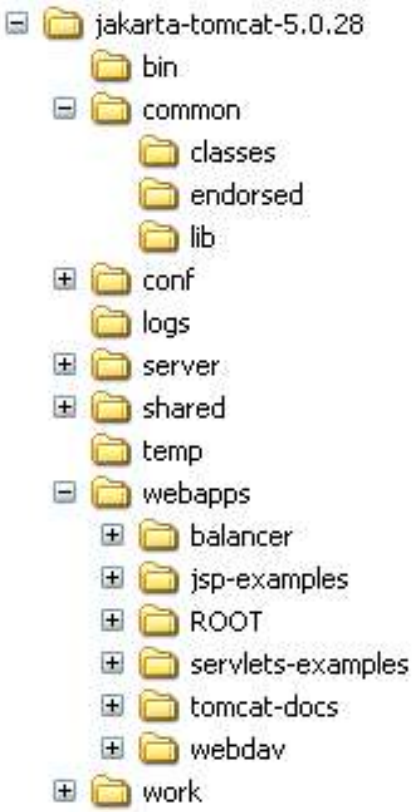

Figura 9 - Estrutura de diretórios do Tomcat

Alguns diretórios destacados na Figura 9 são de extrema importância na configuração e arquitetura do Tomcat, no qual destacamos os seguintes diretórios:

- *Pasta bin*: Contém os scripts e executáveis necessários para a inicialização do container, como o startup.bat e startup.sh, e para a finalização do serviço, como o shutdown.bat e shutdown.sh.
- *Pasta conf*: Nesta pasta estão presentes os arquivos para a configuração do Tomcat, no qual, server.xml contém a definição de contextos, tomcat-users.xml contém a definição dos usuários e permissões e, por fim, web.xml que contém a habilitação do servlet invoker para quem quer definir contextos.
- *Pasta logs*: São armazenados os arquivos de log gerados pelo Tomcat e dos contextos implantados.

- *Pasta webapps*: Local padrão para implantação de aplicações WEB. Nesta pasta, cada aplicação possui sua própria pasta.

Na Figura 10 é mostrado o servidor Tomcat iniciado e o tempo de execução para sua inicialização em milisegundos representado pelo número 22687.

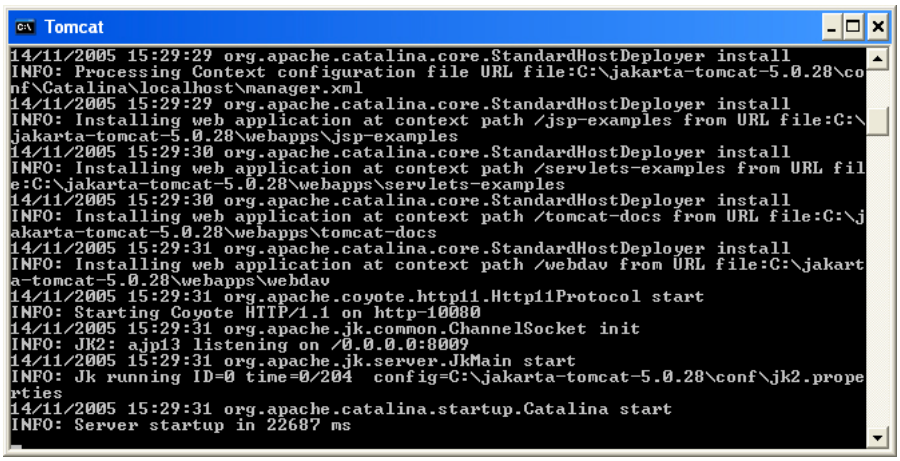

Figura 10 - Start do Tomcat

Para o desenvolvimento da aplicação e do melhor funcionamento do Tomcat, foram realizadas algumas modificações no arquivo de configurações do Tomcat.

No arquivo server.xml foram alterados os parâmetros da porta no qual o navegador requisitará a conexão com o Tomcat, conforme mostrado na Figura 11 e foi inserido o contexto onde servidor WEB buscará os arquivos da aplicação, conforme mostrado na Figura 12

Figura 11 - Configuração da porta

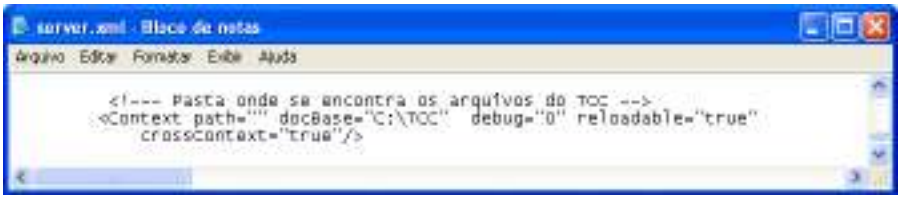

Figura 12 - Configuração da pasta da aplicação

2.7 Considerações Finais

Neste Capítulo foram apresentadas as definições e propriedades da linguagem Java utilizada neste trabalho para a confecção da Ferramenta. No próximo capítulo é apresentada a Ferramenta **jspOraAdmin** e um exemplo de caso de uso para ilustrar as funcionalidades da ferramenta.

3. FERRAMENTA jspOraAdmin

3.1 Arquitetura

Na Figura 13 é apresentada a arquitetura da ferramenta desenvolvida, podendo assim ser visualizado com maior clareza como é feita a requisição do usuário à manipulação do banco de dados Oracle 9i.

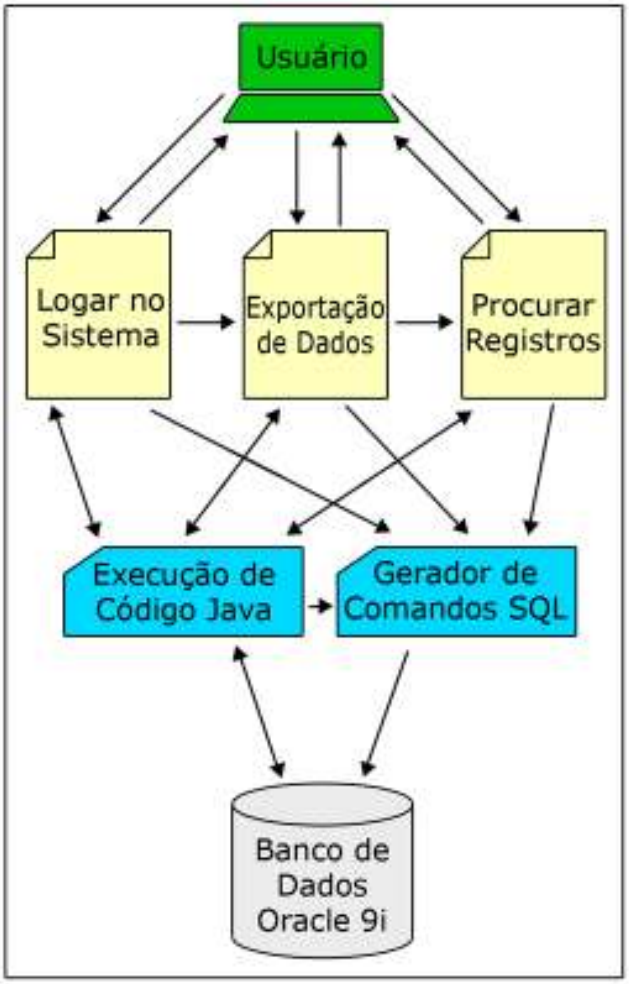

Figura 13 - Arquitetura da Ferramenta jspOraAdmin

A ferramenta consiste de um mecanismo que possibilita a manipulação de dados em um banco de dados Oracle. Para a execução e visualização dos dados são realizadas consultas na base de dados utilizando os dicionários de dados disponíveis. Nos dicionários de dados são armazenados todos os objetos (usuários, tabelas, índices, etc) contidos na base de dados que está sendo manipulada. Em cada tela são gerados códigos Java, assim podendo ser construído o comando SQL para execução no banco de dados.

Para os testes de funcionamento da ferramenta foram utilizados as tabelas padrões do banco de dados Oracle 9i e outros scripts gerados a partir da ferramenta desenvolvida.

No próximo item deste capítulo será abordada as funcionalidades da Ferramenta jspOraAdmin.

3.2 Funcionamento da Ferramenta

Encontra-se neste item a descrição e apresentação das telas de cada elemento da arquitetura da Ferramenta jspOraAdmin.

Login: Na tela inicial da ferramenta, são informados os dados como o HOST onde está localizado (hospedado) o banco de dados, a PORTA que será utilizada para estabelecer a conexão, o USUÁRIO e a SENHA do banco de dados, e o BANCO DE DADOS que será conectado, permitindo assim a manipulação do mesmo. Por ser uma ferramenta WEB, o jspOraAdmin pode ser acessado de qualquer computador ligado à internet, desde que os parâmetros como o IP e a porta sejam informados na URL do navegador. A descrição acima se refere à Figura 14.

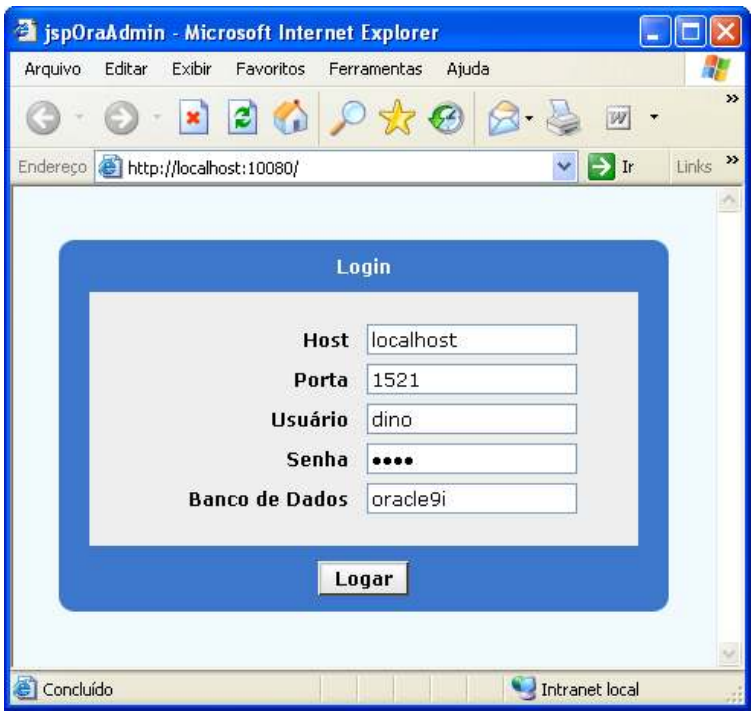

Figura 14 - Tela de login da ferramenta jspOraAdmin

jspOraAdmin: O usuário do sistema tem a opção de escolher entre criar uma nova tabela no banco de dados ou selecionar uma tabela já existente. Se optar em criar uma nova tabela, deve-se informar o nome da tabela e a quantidade de campos que ela terá. Para manipular uma tabela já existente, é necessário selecionar a tabela a ser manipulada no menu à esquerda. No menu, também é informadas a quantidade de função, procedure e view pertencente ao usuário conectado. A descrição acima se refere à Figura 15.

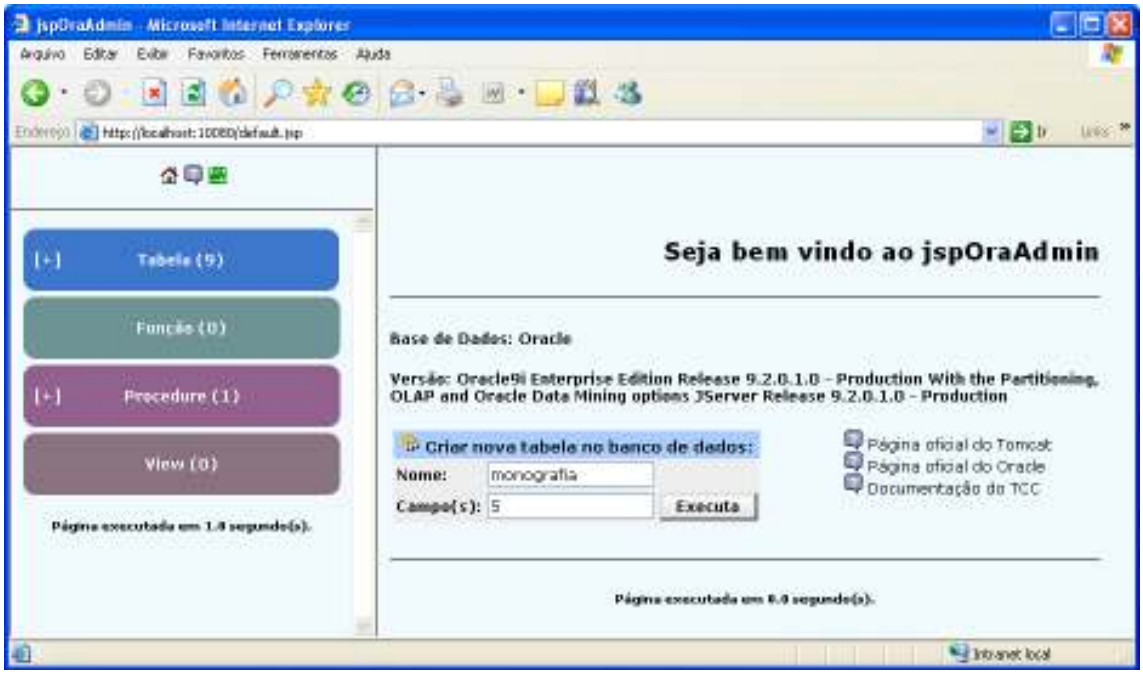

Figura 15 - Tela de abertura e criação da nova tabela

Definições da nova tabela: Esta tela é chamada quando o usuário da ferramenta solicita a criação de uma nova tabela e, para isso, é necessário informar as características do mesmo, que são: nome da coluna, tipo de dados da coluna, tamanho ou valor da coluna, permitir que a coluna seja preenchida de forma obrigatória, selecionar caso deseje que a coluna seja uma chave primária, um índice ou uma chave única. A descrição acima se refere à Figura 16.

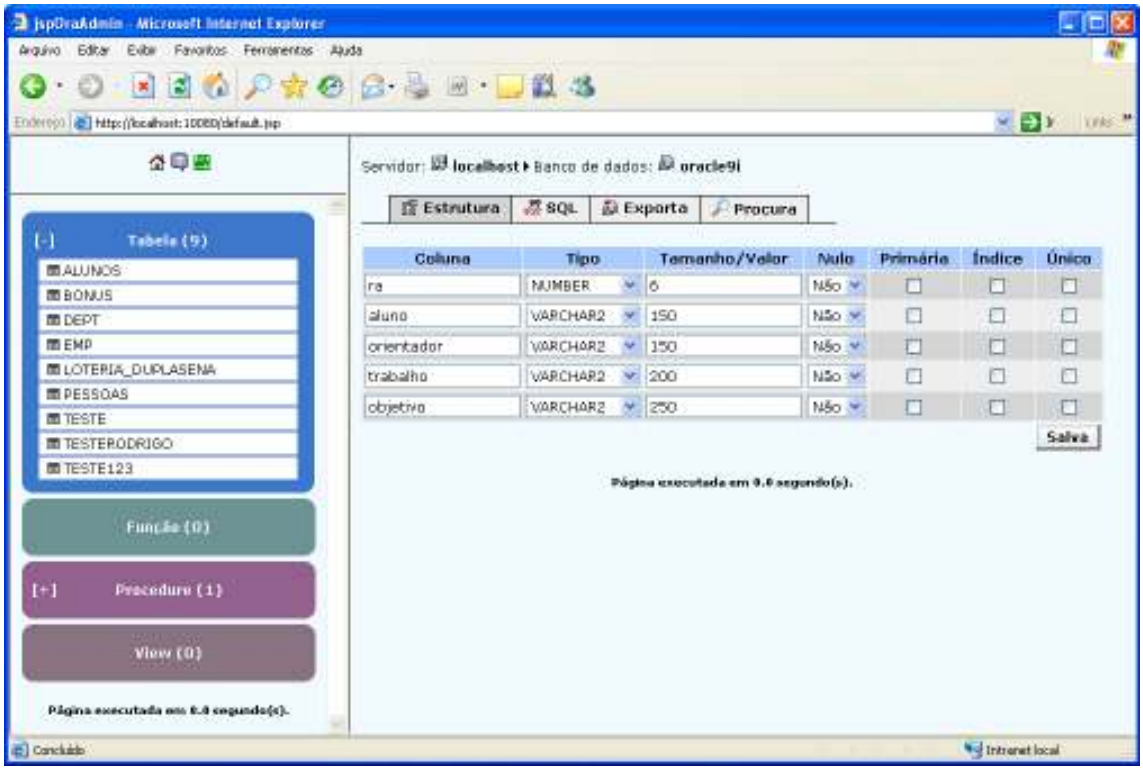

Figura 16 - Definição dos dados para criação da tabela

Estrutura das tabelas: Este é o ambiente onde o usuário visualizará as tabelas criadas e a quantidade de registros inseridos nas respectivas tabelas. Consta nesta tela as ações para manipular uma tabela desejada, que são: visualizar registros, procurar palavra ou valor, inserir registro, visualizar a estrutura, limpar registro e eliminar a tabela. O usuário pode ainda selecionar mais de uma tabela e executar a ação selecionada no combo. Qualquer ação selecionada é exibida uma mensagem de confirmação para que não sejam cometidos erros por engano. A descrição acima se refere à Figura 17.

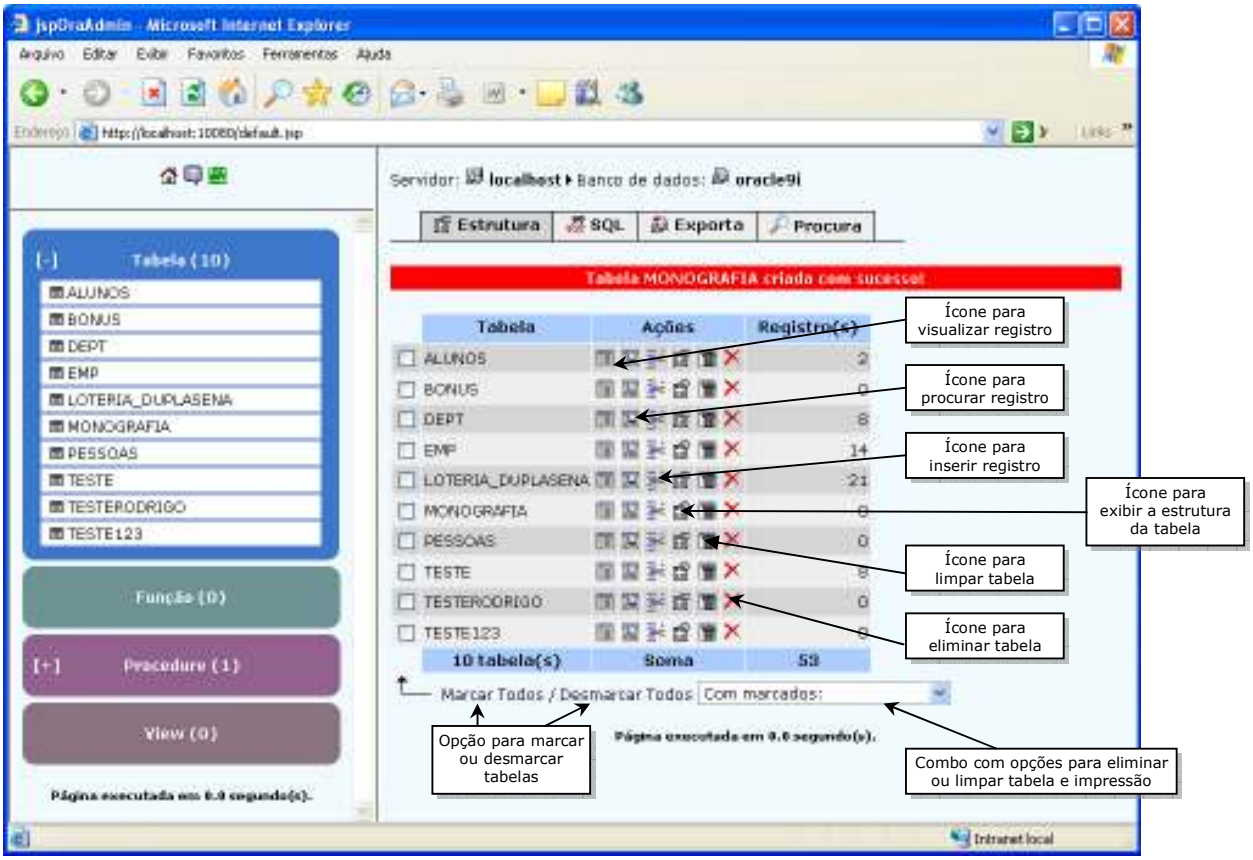

Figura 17 - Estrutura das tabelas do banco de dados

Execução de comandos no banco de dados: Para aqueles usuários mais avançados, que conhecem os comandos do tipo DDL e DML, poderá realizá-lo nesta tela. Será permitido executar somente um comando por vez. Consultas com o comando SELECT estão descartadas desta tela. Caso ocorram erros na execução do comando, será exibido na tela o erro retornado pelo banco de dados. A descrição acima se refere à Figura 18.

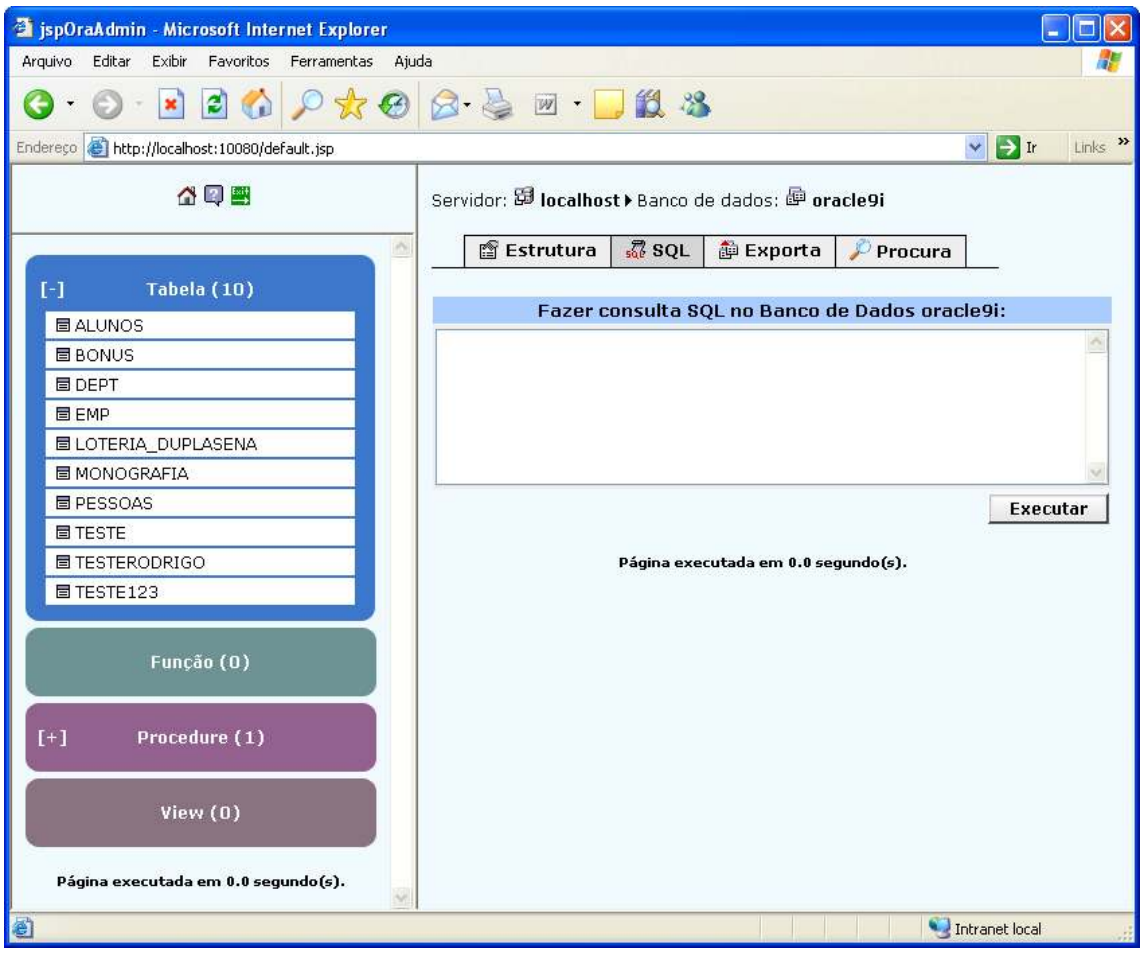

Figura 18 - Execução de comandos SQL no banco de dados

Exportação de dados: No menu Exporta, o usuário terá a opção de exportar as tabelas e suas chaves, e os registros contidos em cada tabela. Os dados serão exportados em forma de comandos SQL, e na seqüência para a execução dos mesmos em outra base de dados Oracle. Esta seqüência é gerada como: estrutura da tabela (colunas, tipo, tamanho e nulo), as chaves (primary, foreign, index e unique), e registros inseridos. A descrição acima se refere à Figura 18 e Figura 20.

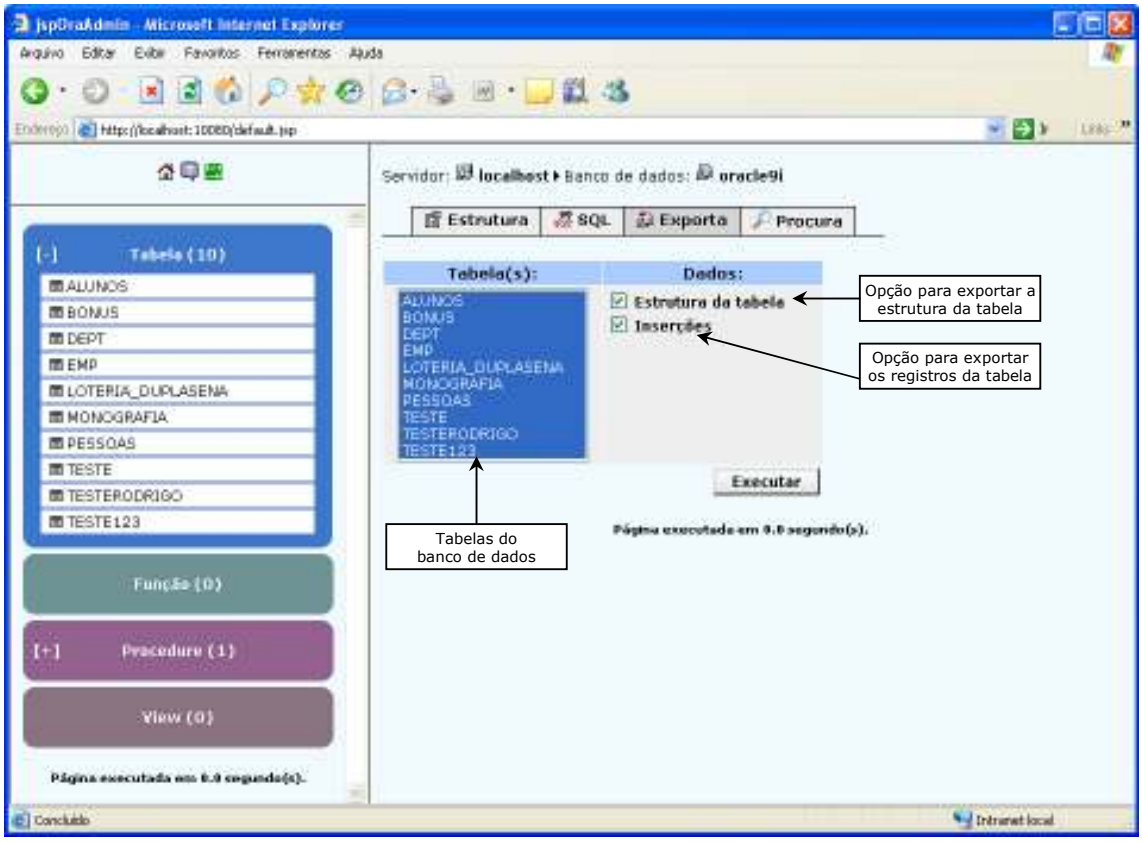

Figura 19 - Exportação dos dados contidos no banco de dados

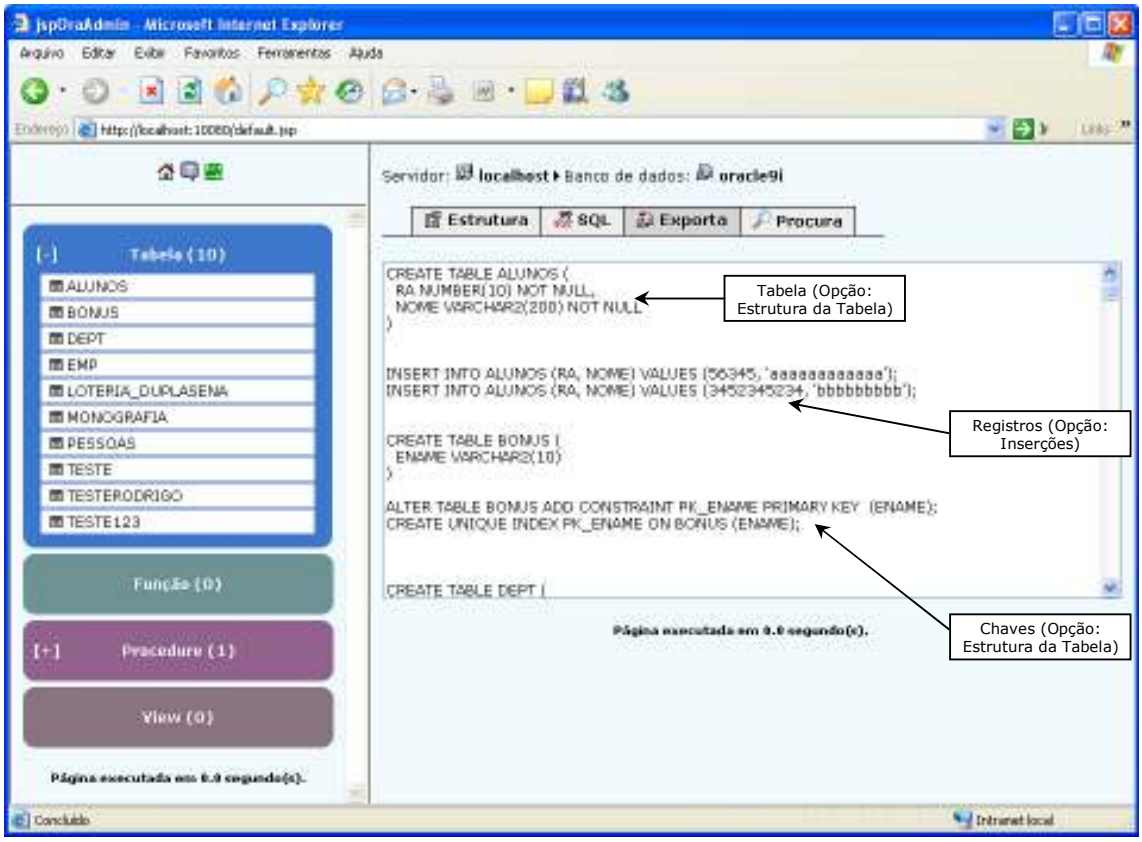

Figura 20 - Dados exportados do banco de dados

Procurar registro no banco dados: Permite que seja procurado palavra(s) ou valor(es) em determinadas tabelas que contenham registros inseridos. O caracter especial "%" representa qualquer cadeia de caracteres. O caracter especial "_" representa um caracter com qualquer valor. Por exemplo, se a procura for realizada com o parâmetro "%a%", o resultado será exibido de acordo com as tabelas selecionadas, mostrando todos os campos que contém a letra "a" em qualquer campo da tabela e em qualquer parte da letra. Com os resultados obtidos, o usuário poderá visualizar os registros encontrados ou removê-los. A descrição acima se refere à Figura 21 e à Figura 22.

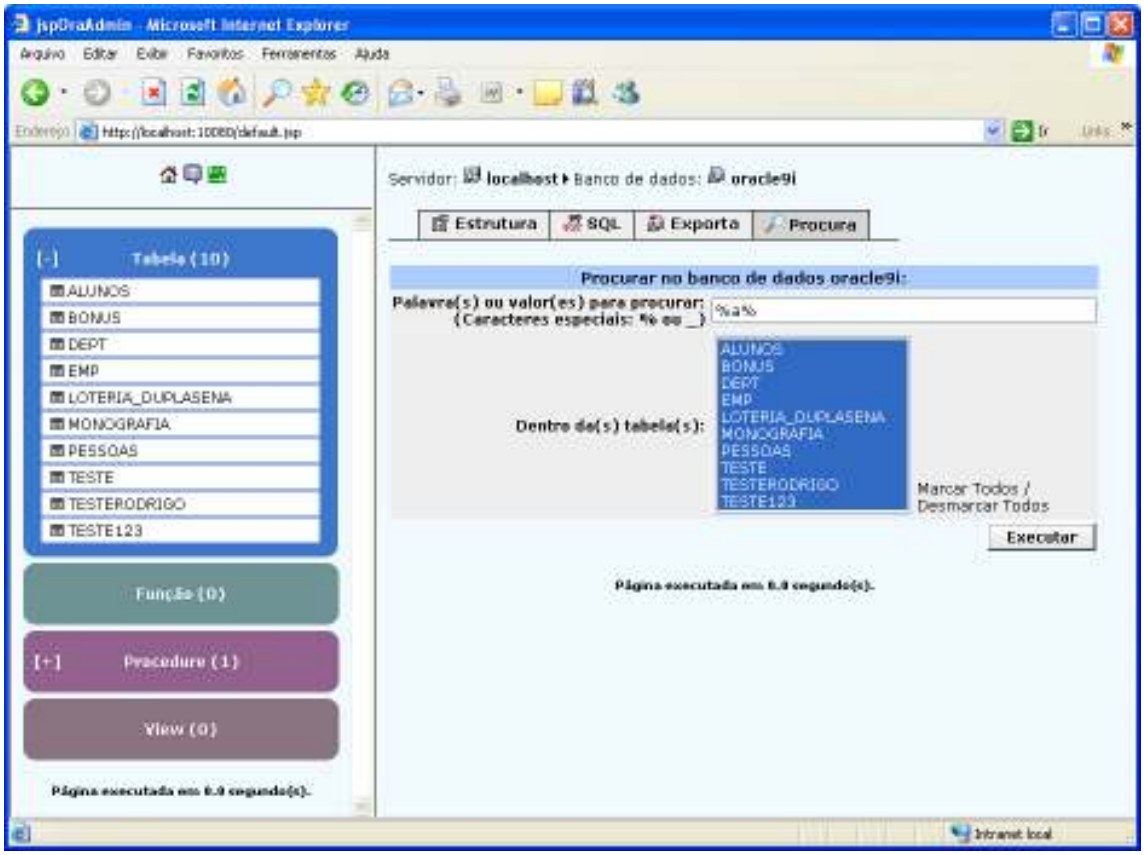

Figura 21 - Procura registros no banco de dados

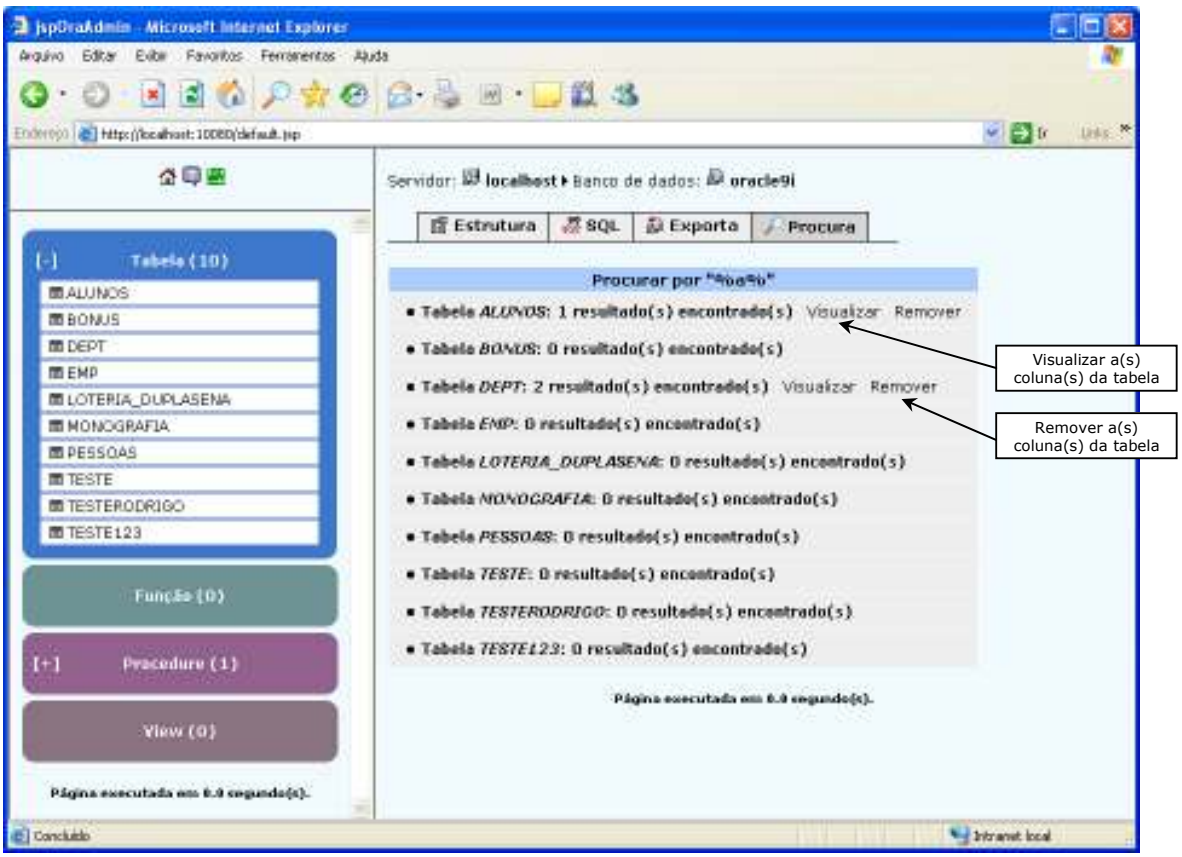

Figura 22 - Registros encontrados no banco de dados

Estrutura das colunas: Neste ambiente, o usuário terá uma visão geral de uma tabela, podendo visualizar as colunas criadas da tabela selecionada e as informações das respectivas colunas, visualizarem ainda as chaves e índices criados, e também criar quantos campos desejados na tabela. Consta nesta tela as ações para manipular uma coluna desejada, que são: alterar coluna, eliminar coluna, criar chave primária, criar índice e criar chave única. O usuário pode ainda selecionar mais de uma coluna e executar a ação selecionada em "Com marcados:". Também é possível eliminar a chave ou o índice desejado. Para criar novos campos na tabela, basta informar a quantidade que campos e executar a ação. Vale ressaltar que qualquer ação selecionada é exibida uma mensagem de confirmação para que não sejam cometidos erros por engano. A descrição acima se refere à Figura 23 e à Figura 24.

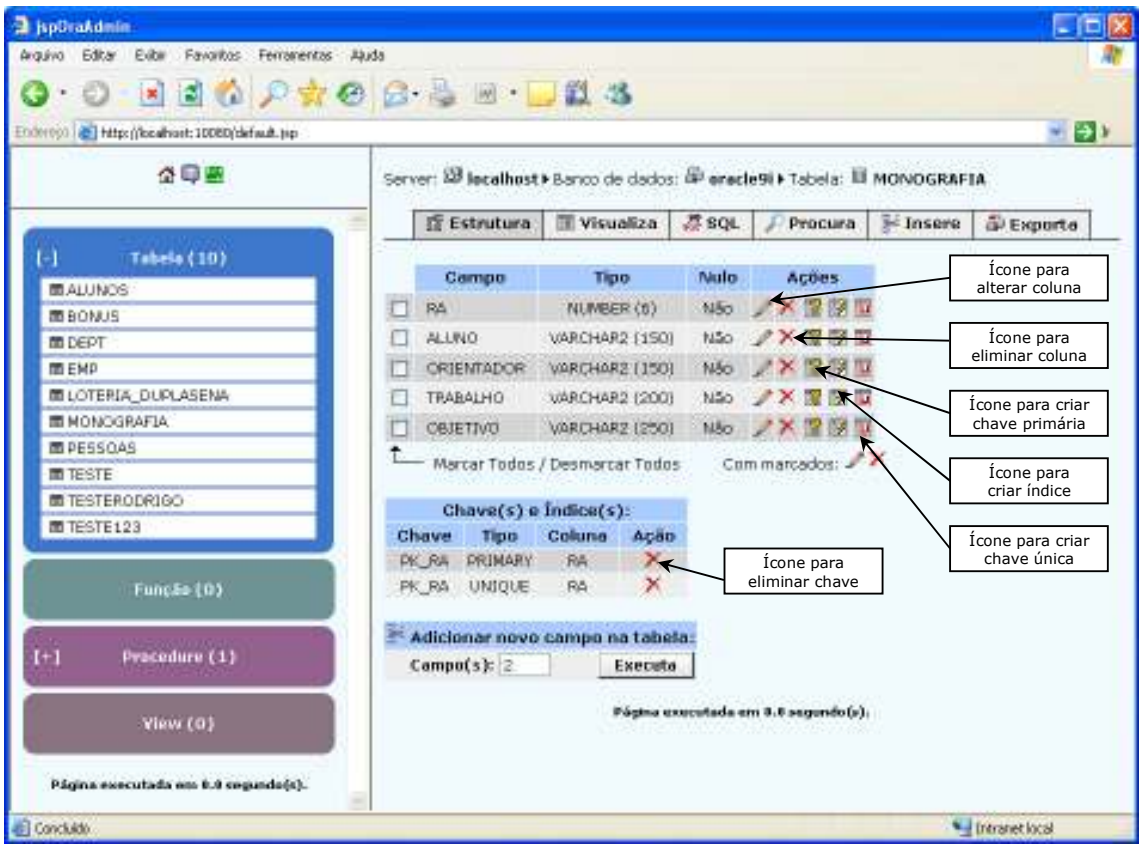

Figura 23- Estrutura das colunas da tabela

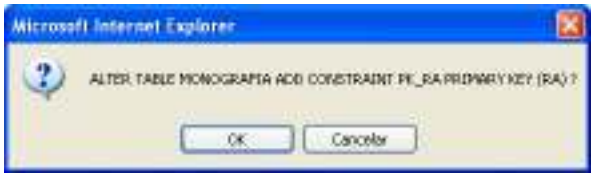

Figura 24 - Confirmação para a criação de uma chave primária

Criação de novos campos: Esta tela é chamada quando o usuário da ferramenta solicita a criação de novos campos na tabela e, para isso, é necessário informar as características do mesmo, que são: nome da coluna, tipo de dados da coluna, tamanho ou valor da coluna, permitir que a coluna seja preenchida de forma obrigatória, selecionar caso deseje que a coluna seja uma chave primária, um índice ou uma chave única. A descrição acima se refere à Figura 25.

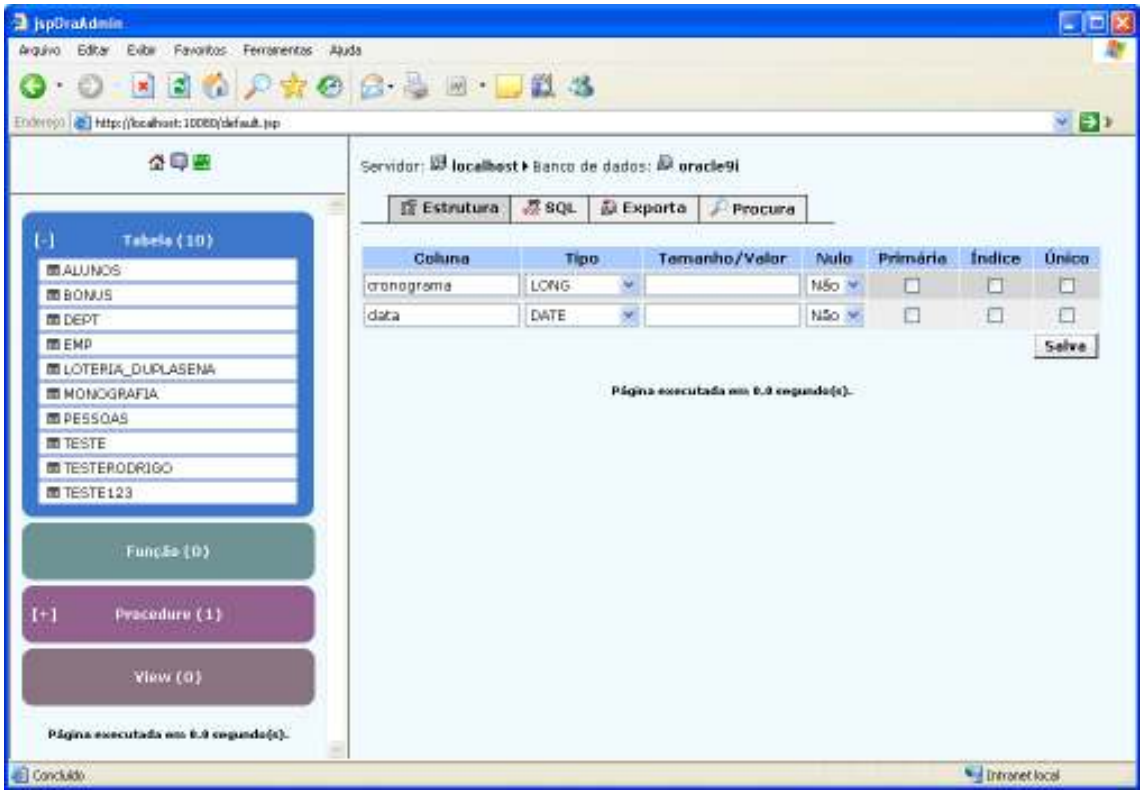

Figura 25 - Criação de novos campos na tabela

Visualização dos registros: No menu Visualiza, serão exibidos todos os registros contidos na tabela, permitindo que o usuário altere ou elimine qualquer registro. Os registros poderão ser alterados ou eliminados selecionando apenas um ou vários registros. Para evitar uma sobrecarga no servidor, esta tela foi configurada para exibir 30 registros por página, e ainda, o usuário poderá ordenar o resultado clicando no título do campo desejado. A descrição acima se refere à Figura 26.

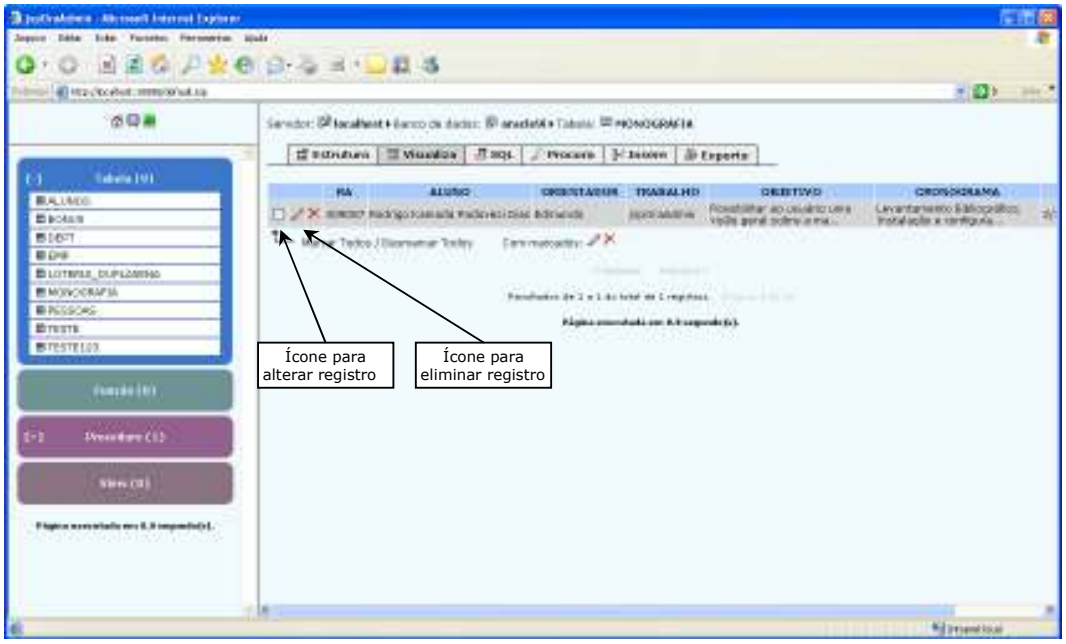

Figura 26 - Visualização dos registros da tabela

Execução de comandos na tabela: Veja descrição no item "Execução de comandos no banco de dados" representado pela Figura 18. A descrição comentada acima se refere à Figura 27.

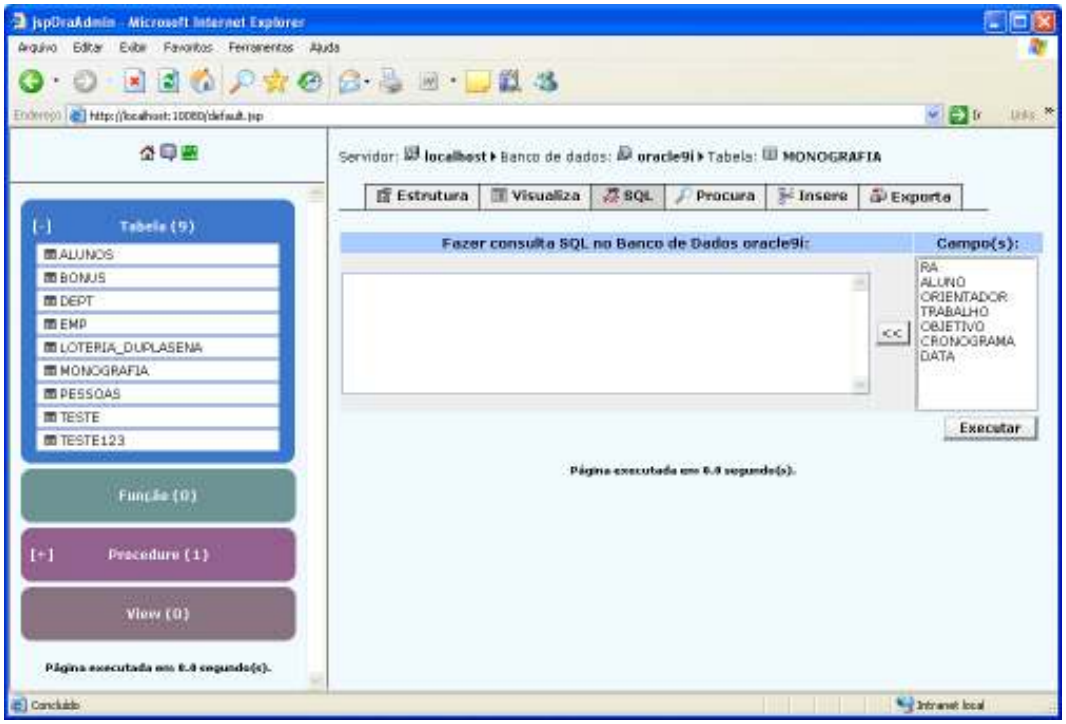

Figura 27 - Execução de comandos SQL na tabela

Procurar registro na tabela: Permite que seja procurada palavra(s) ou valor(es) na tabela que está sendo manipulada que contenham registros inseridos. O usuário terá duas opções para procurar. Na primeira opção, serão escolhidas as colunas, o operador a ser utilizado e digitado a palavra ou valor para assim executar a ação. Já na segunda opção, o usuário informará a condição completa dentro da cláusula WHERE. O resultado será exibido de acordo com os parâmetros informados na consulta, e a tela será igual da Figura 26. A descrição acima se refere à Figura 28.

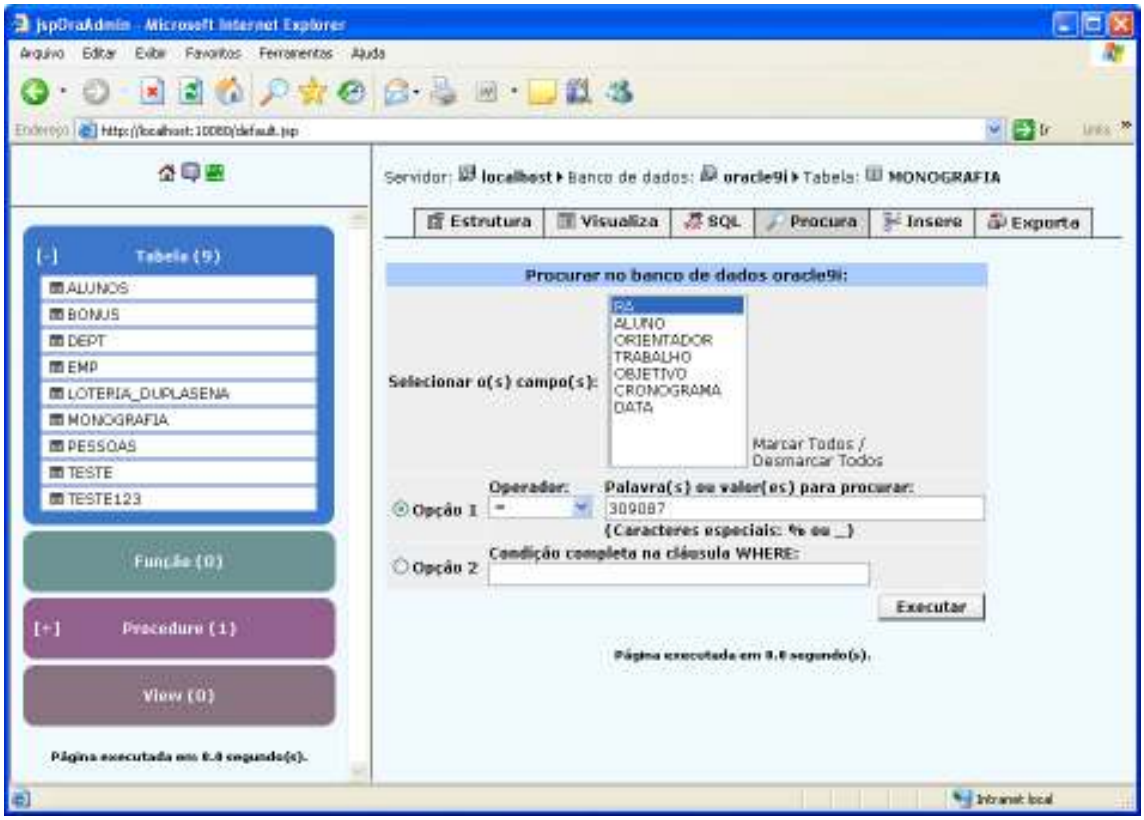

Figura 28 - Procura registros na tabela

Insere registro: Esta tela é chamada sempre que o usuário solicita a alteração de algum registro ou quando vai ao menu Insere para a inserção de registro. Para ajudar o usuário na manipulação desta tela, é informado o tipo do campo, se o campo é nulo e também é limitado

o espaço para a digitação nos campos de acordo com o tamanho da coluna. A descrição acima se refere à Figura 29.

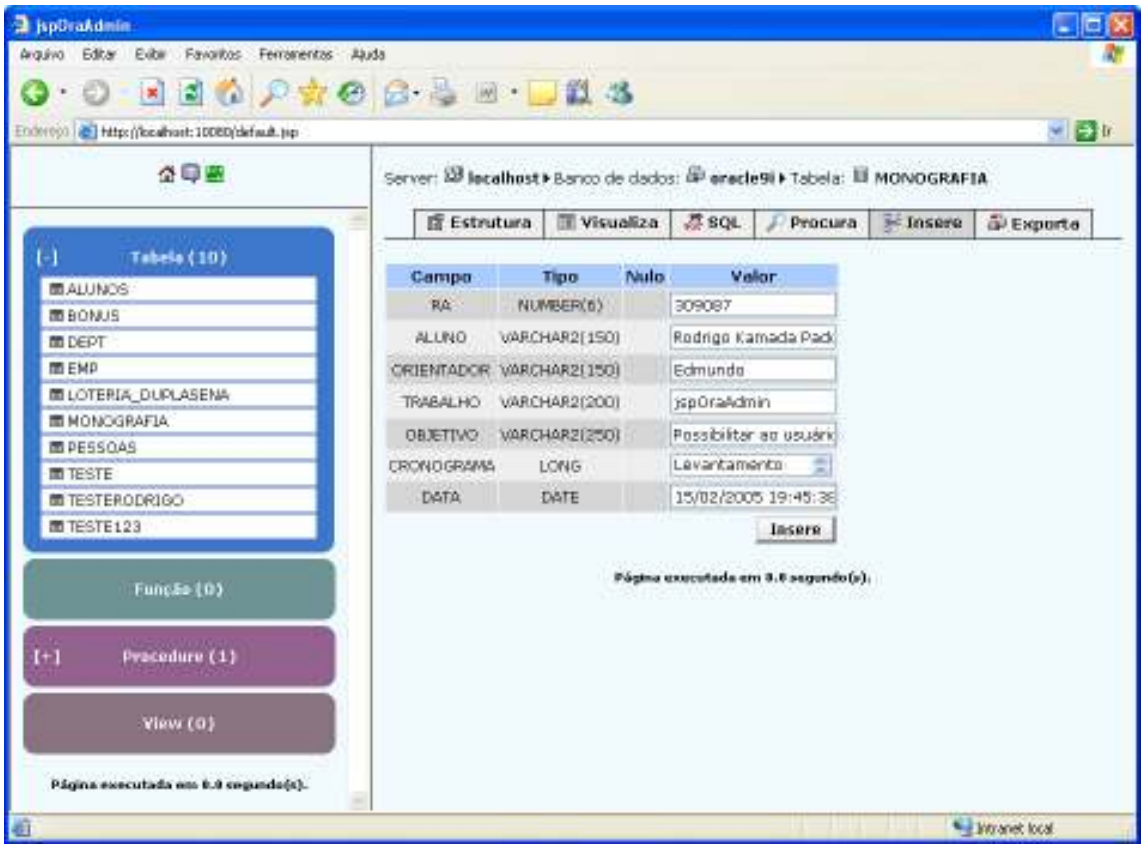

Figura 29 - Insere registros na tabela

4. CONCLUSÃO

Foram estudados o banco de dados Oracle 9i e a linguagem de programação JSP com o intuito de prepará-los como base para o desenvolvimento de uma aplicação voltada para Internet e, a ferramenta adotada como modelo e referência para auxiliar nos estudos foi a jspOraAdmin, detalhada no Capítulo 3 desta monografia.

A linguagem JSP veio para revolucionar o desenvolvimento de aplicações para WEB por ser robusta, facilidade na codificação e permite que o conteúdo possa ser reutilizado. Além disso, programadores e *webdesigners* podem trabalhar juntos, pois permite separação na codificação de programas.

Assim sendo, essa aplicação pode ser usada como um guia rápido ou como um manual de aprendizagem.

Durante o processo de aprendizagem, instalação, configuração e testes realizados pode-se observar que o Oracle 9i é um banco de dados confiável e grande capacidade de gerenciamento dos dados. Outra vantagem é o fato de ser um banco de dados completo, inclusive com a presença de *stored procedures*, *triggers*, integridade referencial, etc.

Conclui-se, portanto que os objetivos propostos no início dessa pesquisa foram alcançados a contento com a ferramenta desenvolvida.

Como proposta para trabalhos futuros, poderiam ser exploradas características para manipulação de funções e *procedures*, e implementação de *triggers*.

Poderiam ser realizadas também pesquisas no sentido de testes de segurança e escalabilidade, distribuição de bancos de dados, etc.

A realização de mais essas pesquisas poderia enriquecer ainda mais o material disponibilizado neste trabalho.

5. REFERÊNCIA BIBLIOGRÁFICAS

APACHE SOFTWARE FOUNDATION. **The Apache Jakarta Tomcat 5 Servlet/JSP Container**. Documentation Index. Disponível em <http://tomcat.apache.org/tomcat-5.0-doc/index.html>. Acesso em: 09 de junho de 2005.

COUCHMAN, J. S., MARISETTI, S. N. **OCP Oracle9i Database: Fundamentos I**. Rio de Janeiro, Ed. Alta Books, 2003.

FIELDS, D. K., KOLB, M. A. **Desenvolvendo na Web com JavaServer Pages**. Rio de Janeiro, Ed. Ciência Moderna Ltda., 2000.

HÓLM, B., CARNELL, et al. **Oracle 9i Java Programando**. Rio de Janeiro, Ed. Alta Books, 2002.