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

RICARDO FILIPE GUIMARÃES GONÇALVES PAULINO

PLEASY: FERRAMENTA DE APOIO AO DESENVOLVEDOR PARA SGBD ORACLE

MARÍLIA
2006

RICARDO FILIPE GUIMARÃES GONÇALVES PAULINO

PLEASY: FERRAMENTA DE APOIO AO DESENVOLVEDOR PARA SGBD ORACLE

Monografia de Trabalho de Conclusão do Curso de Ciência da Computação do Centro Universitário Eurípides de Marília, mantido pela Fundação de Ensino Eurípides Soares da Rocha, para obtenção do Título de Bacharel em Ciência da Computação.

Orientadora:
Prof. Dra. Fátima L. S. Nunes Marques

MARÍLIA

2006

Dedico este trabalho a todos os que me apoiaram na sua execução.

AGRADECIMENTOS

Agradeço o apoio da esposa e família, pelos dias nos quais não pude atendê-los como gostaria de fazê-lo. Agradeço a oportunidade de realizar este trabalho à professora Fátima e aos professores do curso de Ciência da Computação.

Epígrafe

"Não importa quão difícil tenha sido o teu passado, poderás sempre recomeçar hoje."

Buda

PAULINO, Ricardo Filipe Guimarães Gonçalves – **PLEASY**: FERRAMENTA DE APOIO AO DESENVOLVEDOR DE SCRIPTS PARA ORACLE. 2006. 49 F. Trabalho de Conclusão do Curso de Ciência da Computação – Monografia - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2006.

RESUMO

O sistema PLEasy é uma ferramenta de apoio ao desenvolvimento e vizualização para usuários do SGBD Oracle. É um sistema que visa a propiciar algumas facilidades ao administrador de banco de dados, fornecendo um ambiente integrado de desenvolvimento com três recursos: árvore de objetos do banco com tabelas, campos, índices, gatilhos, procedimentos armazenados e funções; editor de textos com funcionalidades como copiar, colar, recortar, alterar a fonte e outros; executor de *scripts* que trabalhe com processamento concorrente utilizando *threads*. É utilizado o dicionário de dados do SGBD Oracle para extrair as informações dos objetos, permitindo ao usuário ver o conteúdo da definição das estruturas de tabelas, procedimentos, funções, gatilhos e outros objetos do banco.

Palavras Chave: Dicionário de Dados, Meta dados, Ferramentas de geração de scripts, Oracle, PLEASY.

PAULINO, Ricardo Filipe Guimarães Gonçalves – PLEASY: FERRAMENTA DE APOIO AO DESENVOLVEDOR DE SCRIPTS PARA ORACLE. 2006. 49 F. Trabalho de Conclusão do Curso de Ciência da Computação – Monografia - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2006.

ABSTRACT

PLEASY is a tool for development support of scripts and includes visualizations for the Oracle DBMS user. It is a system that intend to provide some facilities to the database administrator, bringing him an integrated environment within with three resources: a database object tree with tables, indexes, fields, triggers, stored procedures and functions; a text editor with some functionalities in it like: copy, paste, cut, edit the font type and others; script executer that can work with competitive tasks using *threads*. The tool uses the metadata database to extract the objects information, providing to the user a way to see the structure definitions of tables, procedures, functions, triggers and other database objects..

Keywords: Data dictionary, Meta data, Scripts generating tools, Oracle, PLEasy

ÍNDICE

ÍNDICE DE FIGURAS

LISTA DE ABREVIATURAS

ADO – *ActiveX Data Object*

API - *Application Programming Interface*

DDL - *Data Definition Language*

DLL - *Dynamically Linked Library*

DML - *Data Definition Language*

MDI – *Multi Document Interface*

RAD – *Rapid Application Development*

SGA - *Shared Global Area*

SGBD - *Sistema Gerenciador de Bancos de Dados*

SQL - *Structured Query Language*

INTRODUÇÃO

Uma das dificuldades existentes no desenvolvimento em bancos de dados é a visualização dos objetos envolvidos na criação e manipulação de dados. Isso torna lenta a construção de novos objetos, pois o usuário precisa conhecer a estrutura do banco de dados, que pode ser muito complexa em alguns casos.

Visualização é o termo utilizado para se referir aos mecanismos que permitem a extração de informações de conjuntos complexos de dados. É comum utilizar-se técnicas de computação gráfica, processamento de imagem e realidade virtual para implementar visualizações de dados (BERTI, 2004).

Por meio da visualização é possível utilizar as tecnologias de processamento disponíveis nos computadores mais modernos para oferecer uma percepção rápida de conjuntos de dados (BERTI, 2004).

A visualização usa metáforas como abstração da informação. A escolha de uma metáfora de visualização adequada ao SGBD é fundamental para facilitar o entendimento sobre os dados.

Objetivos do trabalho

Este trabalho representa através de técnicas de visualização da informação estruturas, dados e outros objetos do SGBD de modo a auxiliar programadores no reconhecimento de conjuntos de dados e estruturas do SGBD *Oracle*. Para tanto, foram realizadas pesquisas sobre visualização de dados, sobre o dicionário de dados do SGBD *Oracle* e a construção de objetos gráficos no ambiente *RAD Delphi*.

A ferramenta PLEASY construída neste trabalho apresenta mecanismos de recuperação de informações com o fim de tornar clara a análise de estruturas e dados usando técnicas de visualização.

Disposição do trabalho

Além desta introdução, esta monografia contém os seguintes capítulos:

Capítulo 2 – Comenta sobre as formas de interagir Sistemas Gerenciadores e Visualização, Sistemas de Arquivos, Sistemas Gerenciadores de Bancos de Dados, Modelo Relacional, Visualização da Informação, Dicionário de Dados do SGBD *Oracle* e Trabalhos Correlatos.

- Capítulo 3 – Apresenta a implementação da ferramenta *PLEasy;*
- Capítulo 4 – Apresenta dois estudos de caso com a finalidade de mostrar as funcionalidades da ferramenta.;

Por fim, o último Capítulo – Apresenta as conclusões e, ao final, são disponibilizadas as referências bibliográficas utilizadas para fornecer o embasamento teórico da monografia.

2 SGBD(s) e Visualização de Informação

Existem várias maneiras de se representar dados. Com a visualização vários áreas no desenvolvimento de softwares se beneficiaram. Entre os vários tipos de visualizações podem ser destacadas a científica, a volumétrica, a médica, a geográfica e a de informações abstratas (BERTI, 2004).

"Visualizações baseadas em mapas geográficos são simples, intuitivas e até naturais. Elas quebram a barreira entre um sistema complexo e o conhecimento de uma área de domínio" (CHEN, 1999).

De fato, as informações organizadas sobre estruturas geográficas são bastante intuitivas o que as torna mais compreensíveis, porém isto não ocorre em todo tipo de representação, Chen (1999) relata questões relacionadas à dificuldade de compreensão de dados abstratos comparando-os a visualização geográfica. Como não existe um padrão específico a ser adotado para representarem dados através de uma visualização, ocorrem, em conseqüência disso, certas dificuldades durante a escolha de uma representação adequada (BERTI, 2004).

Com o avanço da tecnologia de armazenamento, bancos de dados tornaram-se maiores e mais complexos, imagens, dados e objetos de diferentes mídias têm sido guardados e recuperados usando uma nova tecnologia: os sistemas gerenciadores de banco de dados, também chamados de SGBD(s) (BERTI, 2004).

A sobrecarga de informações é uma das principais preocupações na representação de resultados obtidos por meio de mecanismos de recuperação de informações. Uma abordagem

para contornar as dificuldades de selecionar as informações relevantes dentre o resultado de buscas é utilizar técnicas de visualização SGBD(s) (BERTI, 2004).

Para que o volume e a complexidade tão grande dos SGBD(s) possam ser assimilados pelo ser humano é necessária a utilização de mecanismos visuais para representar conjuntos de dados por meio de abstrações, ou seja, metáforas (BERTI, 2004).

Por meio dessas técnicas o usuário pode obter uma representação visual, que, se por um lado abstrai detalhes do conjunto de informações, por outro, proporciona a organização desse conjunto segundo algum critério.

2.1 Sistemas de arquivos

Segundo Harrington (2002) os primeiros sistemas de processamento de arquivo eram compostos por um conjunto de arquivos de dados, mais comumente arquivos de texto.

As aplicações que usassem sistemas de arquivos deviam realizar todas as operações necessárias tais como inserção, exclusão, organização das tabelas de registros em diferentes ordens como crescente e decrescente. Se fosse necessário, por exemplo, filtrar dados, isso era feito manualmente.

Harrington (2002), discutindo sobre arquivos de dados, diz que esse tipo de arquivo é organizado em um formato fixo e muito preciso.

De acordo com Harrington (2002) os sistemas realizados com esse modelo de armazenamento em arquivos de dados estão sujeitos alguns problemas, por exemplo:

- A alteração do *layout* de um arquivo de dados (por exemplo, a alteração do tamanho de um campo) requer a alteração de todos os programas que fazem acesso a esse arquivo como também a regravação de todos os dados desse arquivo, pois uma eventual pesquisa para um registro inserido antes da alteração do *layout* do arquivo, resultaria em informações equivocadas (HARRINGTON, 2002).

- A pesquisa de dados específicos neste tipo de estrutura pode se tornar demasiadamente lenta visto que a pesquisa seria seqüencial e deveria percorrer todos os bytes até chegar-se à informação desejada (HARRINGTON, 2002).

- A principal vantagem de um sistema de processamento de arquivos é que ele é simples e mais barato, porém, um sistema deste tipo fica sujeito a inconsistências como, por exemplo, a redundância desnecessária de dados que provoca, inevitavelmente incongruência de dados (HARRINGTON, 2002).

Devido a esses problemas surgiu a necessidade de desenvolver softwares que facilitassem a tarefa de manipulação de dados, denominados Sistemas Gerenciadores de Bancos de Dados (SGBD).

2.2 Sistemas Gerenciadores de Bancos de Dados

Segundo Machado (2004), um banco de dados pode ser definido como um conjunto de dados devidamente relacionados. Pode-se compreender como dados os objetos conhecidos que podem ser relacionados e que possuem significado implícito. Porém o significado do termo banco de dados é mais restrito do que a definição dada anteriormente.

Um banco de dados possui as seguintes características:

- É uma coleção lógica coerente de dados com um significado inerente; uma disposição desordenada dos dados não pode ser referenciada como um banco de dados (MACHADO, 2004).
- É projetado, construído e populado com valores para um propósito específico; um banco de dados possui um conjunto pré-definido de usuários e aplicações (MACHADO, 2004).
- Representa algum aspecto do mundo real, o qual é chamado de minimundo, qualquer alteração efetuada no minimundo é automaticamente refletida no banco de dados (MACHADO, 2004).

Conforme afirmam Korth, Silberchatz e Sudarshan (1999) um SGBD é constituído por um conjunto de dados associados a um conjunto de programas para acesso a esses dados.

A intenção no desenvolvimento de banco de dados é gerir um negócio de maneira mais eficiente, rapidamente e com segurança. Em SGBD(s), dados são inter-relacionados para garantir integridade e eficiência na realização de pesquisas. O SGBD fornece estruturas, regras e funções para facilitar essa implementação (MACHADO, 2004).

Na definição de estruturas para os dados usa-se a Linguagem chamada DDL (*Data Definition Language*), para a manipulação de tabelas existe a linguagem DML (*Data Manipulation Language*) (MACHADO, 2004).

2.3 Modelo Relacional

De acordo com Machado (2004) um novo conceito de banco de dados começou a ser usado e implementado nas linguagens no final dos anos 80 com a proposta do modelo relacional.

Esse modelo foi criado no final dos anos 70 por "Edgar F. Codd" e, somente, começou a ser utilizado por um número maior de empresas por volta de 1987. Esse modelo se refere às estruturas de dados como relações matemáticas. Este é um modelo lógico que deu origem à teoria relacional (MACHADO, 2004).

A teoria relacional estuda os conceitos lógicos dos bancos de dados relacionais e nela estabeleceram-se várias regras para descrever a tabela relacional. Em 1979 Edgar F. Codd e Chris Date refinaram o modelo relacional dando origem ao que se chama hoje de modelo relacional estendido (MACHADO, 2004).

A visão utilizada pela teoria relacional é de que o banco de dados é um conjunto de tabelas com linhas e colunas como mostra o exemplo da Figura 2.1 A intenção dessa forma de tratar os dados é fornecer transparência aos usuários, programadores ou não, de forma que não seja necessário que se saiba onde e como estão os dados do sistema. (MACHADO, 2004)

Registro	Código	Nome	Idade	Sexo
1	10	Luis	58	Masculino
2	20	Ricardo	28	Masculino
3	30	Vilma	31	Feminino
4	40	Delfina	56	Feminino

Figura 2.1 - Tabela

Foram estabelecidos critérios para a teoria relacional:

- Cada uma das tabelas é chamada de relação;
- O conjunto de uma linha de colunas é chamado de tupla;
- Cada coluna tem um nome específico e representa um domínio diferente;
- Não há duas linhas iguais;
- Cada tabela tem um nome único no Banco de Dados;
- A ordem das linhas e das colunas não é importante.

Algumas das principais operações para se dar manutenção em um Banco de Dados serão apresentadas a seguir.

Para realizar pesquisas no banco de dados usa–se uma linguagem chamada *SQL.* Machado (2004) lembra que SQL só previa o seu uso de forma interativa. Após sofrer alguns acréscimos, ela passou também a ter capacidade de ser utilizada em linguagens hospedeiras, tais como *Cobol, Fortran*, *"C"*.

Machado (2004) afirma que a Linguagem *SQL* assume um papel muito importante nos SGBD(s), podendo ter muitos enfoques:

- **Linguagem Interativa de Consulta** (*Query AdHoc*) – com os comandos *SQL* os usuários podem montar consultas poderosas sem a necessidade de criar um programa podendo utilizar *forms* ou ferramentas de montagem de relatório (MACHADO, 2004).

- **Linguagem de Programação para Acesso a Banco de Dados** – comandos *SQL* embutidos em programas de aplicação que acessam os dados armazenados (MACHADO, 2004).

- **Linguagem de Administração de Bancos de Dados** – responsável pela administração do banco de dados (*Data Base Administrator*) pode utilizar comandos *SQL* para realizar suas tarefas (MACHADO, 2004).

- **Linguagem Cliente/Servidor** – os programas (Cliente) dos computadores pessoais usam comandos *SQL* para se comunicarem por meio de uma rede local, compartilhando os dados armazenados em um único local (Servidor). A arquitetura cliente/servidor minimiza o tráfego de dados pela rede (MACHADO, 2004).

- **Linguagem para bancos de dados distribuídos** – a *SQL* auxilia na distribuição dos dados através de vários nós conectados ao sistema de aplicação. Auxilia também na comunicação de dados com outros sistemas (MACHADO, 2004).

- **Caminho de acesso a outros bancos de dados em diferentes máquinas** - a *SQL* auxilia na conversão entre diferentes produtos de banco de dados colocados em diversas máquinas (de micro até *mainframes*) (MACHADO, 2004).

- **Definição de dados (DDL)** – permite ao usuário a definição da estrutura e organização dos dados armazenados e as relações entre eles (MACHADO, 2004).

- **Manipulação de dados (DML)** – permite ao usuário ou a um programa de aplicação a inclusão, remoção, seleção ou atualização de dados previamente armazenados no banco (MACHADO, 2004).

- **Controle de acesso** – protege os dados de manipulações não autorizadas (MACHADO, 2004).

- **Compartilhamento de dados** – coordena o compartilhamento dos dados por usuários concorrentes, sem, contudo, interferir na ação de cada um deles (MACHADO, 2004).

- **Integridade dos dados** – auxilia no processo de definição da integridade dos dados, protegendo contra corrupções, inconsistências e falhas do sistema de computação (MACHADO, 2004).

2.4 Visualização da Informação

Desde 1786 a necessidade de se representarem dados utilizando imagens tem sido tema para pesquisas, John Playfair (CARD *et al*., 1999) - geólogo e matemático escocês trabalhava com representação de dados gráficos.

O fundamento da Visualização de Informação está diretamente ligado à cognição humana. A representação é associada com essa peculiaridade do ser humano de modo a facilitar sua memorização e compreensão.

O conceito da Visualização de Informação é o uso de representação visual, interativa e com suporte computacional de dados abstratos para ampliar a cognição humana. Através de uma imagem é possível distinguir com um rápido olhar o que se julgava difícil (CARD *et al*., 1999).

2.5 Dicionário de Dados do SGBD Oracle

"O negócio da Oracle é a informação: como gerenciar, usar, compartilhar, proteger. Há quase três décadas a Oracle, maior empresa de software empresarial do mundo, fornece software e serviços que possibilitam às empresas extrair as informações mais precisas e atualizadas de seus sistemas de negócios." (ORACLE, 2006).

O *Oracle* é um SGBD que surgiu no final dos anos 70, quando Larry Ellison vislumbrou uma oportunidade que outras companhias não haviam percebido, quando encontrou uma descrição de um protótipo funcional de um banco de dados relacional e descobriu que nenhuma empresa tinha se empenhado em comercializar essa tecnologia.(WIKIPEDIA, 2006).

Ellison e os co-fundadores da *Oracle Corporation*, Bob Miner e Ed Oates, perceberam que havia um tremendo potencial de negócios no modelo de banco de dados relacional tornando assim a maior empresa de software empresarial do mundo (WIKIPEDIA, 2006).

O SGBD da *Oracle* é líder de mercado. O *Oracle* 9i foi pioneiro no suporte ao modelo *web*. O *Oracle* 10g, mais recente, se baseia na tecnologia de *grid.* Além da base de dados, a Oracle desenvolve uma *suíte* de desenvolvimento chamada de *Oracle Developer Suite*, utilizada na construção de programas de computador que interagem com a sua base de dados. Além disso, a *Oracle* também criou a linguagem de programação PL/SQL, utilizada no processamento de transações (WIKIPEDIA, 2006).

O SGBD Oracle fornece um dicionário de dados, sua função é armazenar as estruturas dos dados. Costuma-se dizer que o "dicionário de dados contém dados sobre os dados do banco", é por meio do dicionário de dados que podemos recuperar informações como, por exemplo, o nome de todas as tabelas contidas em um esquema de usuário ou saber qual o tipo das colunas das tabelas.

O administrador do banco de dados precisa ter conhecimentos sobre como funciona, como recuperar informações úteis que não são visíveis ao usuário comum e até mesmo

reparar danos no BD. Além disso, o dicionário de dados tem um papel importante na manutenção da integridade e segurança do banco de dados.

Ao executar um *script* de procedimento, função, *trigger* ou outros, o interpretador do *Oracle* verifica a sintaxe, os nomes dos objetos, a sua ordem, a seqüência, a escrita dos comandos e a escrita de operadores (ASADI, 1997).

O *Oracle* pode ainda recuperar objetos utilizados nas operações realizadas, mantê-los na área compartilhada do banco de dados (ASADI, 1997).

A principal tabela do dicionário de dados do *Oracle* 9i é a *view DICTIONARY*, que descreve o nome de todas as outras *views* contidas na memória compartilhada do banco. Na Figura 2.2 são mostradas algumas das *views* usadas neste trabalho e sua descrição, no Capítulo 3 são exibidos trechos de código no qual elas são usadas.

Tabelas	Comentário
USER_ALL_TABLES	Descrição de todas os objetos e tabelas relacionais pertencente ao usuário
USER_CONSTRAINTS	Definição de "constraints" nas tabelas pertencentes ao usuário
USER_SOURCE	Conteúdo de objetos armazenados acessíveis ao usuário
USER_TABLES	Descrição das tabelas de usuário
USER_TAB_COLS	Colunas de tabelas, *views* e *clusters*
USER_TAB_COLUMNS	Colunas de tabelas, *views* e *clusters*
USER_TRIGGERS	Triggers (Gatilhos) pertencentes ao usuário
USER_TRIGGER_COLS	Coluna usada nas Triggers (Gatilhos) do usuário
USER_USERS	Informações sobre o usuário corrente
USER_VIEWS	Descrição das views pertencentes ao usuário

Figura 2.2 – Descrição de algumas *views* usadas neste trabalho

2.6 Trabalhos Correlatos

Na literatura são citadas algumas ferramentas construídas para o desenvolvimento de *scripts* em *PL/SQL* e visualização de informação. Entre as ferramentas pesquisadas são apresentados pelo menos um dos seguintes recursos: visualização de objetos do banco, editores de texto para *scripts* e *recuperação de estruturas.*

As ferramentas escolhidas como modelos reconhecem os objetos do SGBD e mostram de alguma forma uma visualização particular do banco de dados e seus objetos.

2.6.1 A Ferramenta FastMapDB

O trabalho de TRAINA *et al.* (2001) apresenta uma ferramenta que disponibiliza de forma gráfica os dados armazenados em uma base relacional.

Essa ferramenta permite que aos usuários participarem do processo de descoberta de conhecimento sobre os dados em análise. O trabalho desenvolve um conceito que suporta a transformação sobre um conjunto de dados permitindo a visualização de atributos de diversos tipos (números, datas e textos). A ferramenta realiza transformações de dados multidimensionais um espaço tridimensional, utilizando uma função de distância definida pelo usuário.

A Ferramenta *FastMap DB* baseia-se na aplicação da técnica *FastMap*, inicialmente desenvolvida com o objetivo de redução de dimensionalidade em espaços de altas dimensões. Na Figura 2.3 é mostrada uma interface da ferramenta

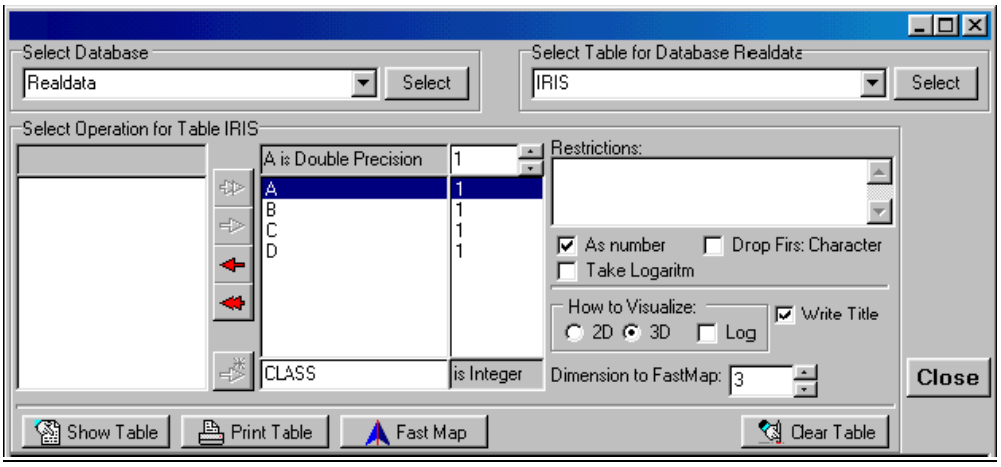

Figura 2.3 - Ferramenta FastMapDB

Na Figura 2.4 é mostrado um experimento detalhado deste trabalho, que visualiza o conjunto dos votos do congresso americano em 1984. Cada registro corresponde ao voto de um congressista em 16 assuntos (por exemplo, gastos com educação, crime, etc.).

Todos os atributos têm o valor 1 (aprova), -1 (não aprova) ou zero (nulo ou abstenção). Cada t-upla tem também um atributo categórico, indicando a que partido, Republicano ou Democrata o congressista correspondente pertence .

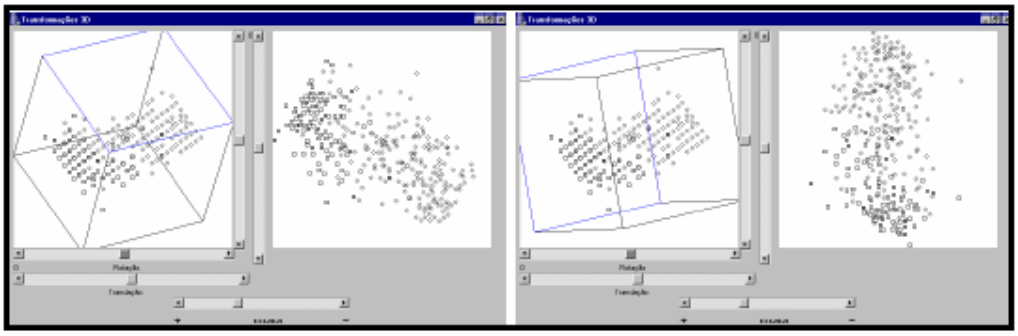

Figura 2.4 – Dois ângulos de visualização do conjunto votos.

2.6.2 GraphMiner: Uma Ferramenta de Mineração Visual de Dados em Bases Relacionais

No trabalho de NETO et al. (2004) foi desenvolvida uma ferramenta chamada *GraphMiner* com o objetivo de apoiar a descoberta de informações úteis em bancos de dados relacionais através da representação gráfica dos objetos do banco na forma de um grafo.

Esse trabalho oferece um mecanismo para a exploração de dados relacionais de forma simples. Além da criação de grafos para representar objetos do banco de dados, esta ferramenta é independente do banco de dados.

Para a construção de um grafo a partir de um banco relacional a ferramenta *GraphMiner* realiza quatro etapas básicas:

1. Escolher uma entidade raiz do banco que se quer visualizar.
2. Descobrir quais são as entidades que se relacionam com a entidade raiz.
3. Selecionar os registros da entidade raiz e de cada uma das entidades relacionadas.
4. Mapear os registros como nós do grafo e as relações entre elas como arestas.

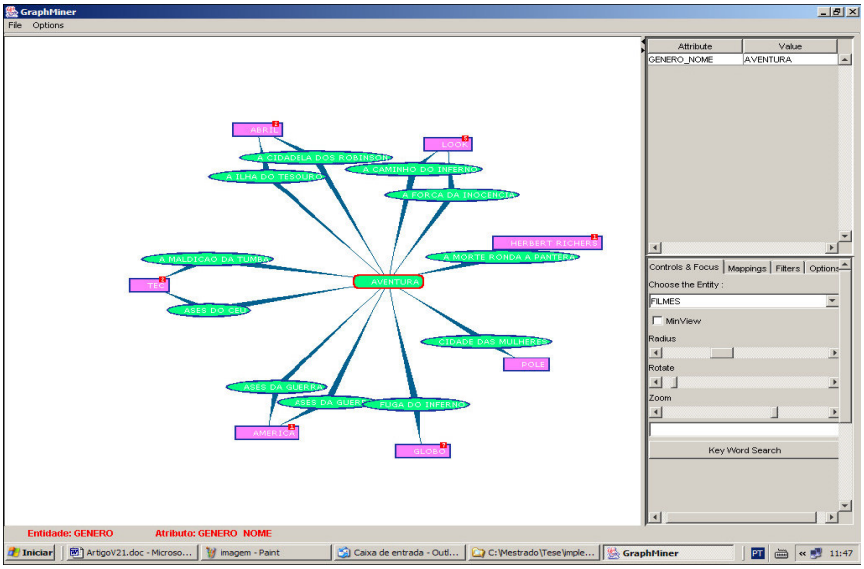

Figura 2.5 - Visualização de dados e suas relações através da *GraphMiner*.

Uma outra funcionalidade que segue a mesma linha do controle e aproximação é o controle do raio de exibição como mostrada na Figura 2.5.

A partir desta funcionalidade a Ferramenta *GraphMine* permite controlar a quantidade de nós exibidos na tela usando como parâmetro para mostrar ou ocultar um nó, a sua distância do nó selecionado. Neste caso, a distância entre dois nós corresponde à quantidade de arestas entre o nó de origem selecionado e nó destino.

A ferramenta disponibiliza ainda os controles de interação que permitem o mapeamento de atributos visuais exibidos no grafo como a cor, o formato e o rótulo dos nós e arestas com qualquer um dos atributos do banco de dados relacional.

2.6.3 VRVis: Ferramenta de Realidade Virtual para Visualização de Informações

O objetivo do trabalho *VRVis* é apresentar uma ferramenta para visualizar grandes volumes de informações abstratas a partir de técnicas de Realidade Virtual com o fim de apresentar a representação da informação de forma intuitiva para o usuário.

Esse trabalho estrutura-se no estudo e utilização de técnicas de Realidade Virtual para Visualização de Informação. Foi usado um banco de dados de imagens de dados médicos com grande volume de informações.

Segundo Berti (2004) grande parte das ferramentas que representam dados de forma gráfica utilizam histogramas ou gráficos de colunas e linhas. Desta forma, uma ferramenta de visualização que utiliza técnicas de Realidade Virtual para demonstrar os dados em consultas pode ter grande contribuição no aspecto da compreensão e percepção das características destes dados.

Na Figura 2.6 é exibida a tela de seleção de parâmetros da consulta da ferramenta de Visualização de Informação VRVis.

Figura 2.6 - Tela de seleção de parâmetros da ferramenta de visualização VRVis.

Por meio da tela apresentada na Figura 2.6, o usuário seleciona os parâmetros para filtragem e composição da consulta.

3 Ferramenta PLEasy

Este capítulo trata da ferramenta construída neste trabalho, versará sobre partes importantes de seu código e funcionamento como classes e métodos. O capítulo também discutirá sobre as tecnologias usadas durante o seu desenvolvimento.

Para se manipular um SGBD é necessário ter conhecimentos sobre suas estruturas e comandos. Em sistemas que usam a linguagem *SQL* como padrão, para criação e manipulação de suas estruturas, o programador precisa estar consciente sobre a versão da linguagem e dos comandos necessários para manipulá-lo. Desta forma, a ferramenta *PLEasy* é um programa implementado em Delphi que fornece funcionalidades e visualizações para SGBD *Oracle,* proporcionando ao programador um ambiente integrado para o desenvolvimento, manipulação de dados e estruturas, e reconhecimento visual de dados através de técnicas de "Visualização da Informação".

Para dar ênfase ao diferencial do programa serão descritas partes de seu código usado para o desenvolvimento da visualização de uma bateria, a árvore de objetos do banco e o Editor/Executor de *scripts*.

3.1 Tecnologias utilizadas

Para implementar o *PLEasy* foram utilizadas diversas tecnologias, entre elas a Ferramenta Delphi, o SGBD *Oracle* e a tecnologia ADO da *MicroSoft.* Foram implementadas basicamente quatro funcionalidades relacionadas à visualização de dados e auxílio ao uso da linguagem SQL: árvore de objetos, editor/executor de comandos, metáfora de bateria e gráfico de barras.

3.2 Árvore de Objetos

A árvore de objetos de banco de dados é uma funcionalidade do programa que mostra inicialmente um nó chamado 'Banco Oracle' de onde derivam os nós 'Tabelas', 'Procedimentos', 'Funções' e 'Gatilhos'. Foi construída usando-se para isto os componentes *TTreeView*, *TImageList* além dos componentes de acesso ao banco de dados. Para se obter os objetos foi usado o dicionário de dados *Oracle*. Na Figura 3.1 é mostrada um diagrama da árvore de dados.

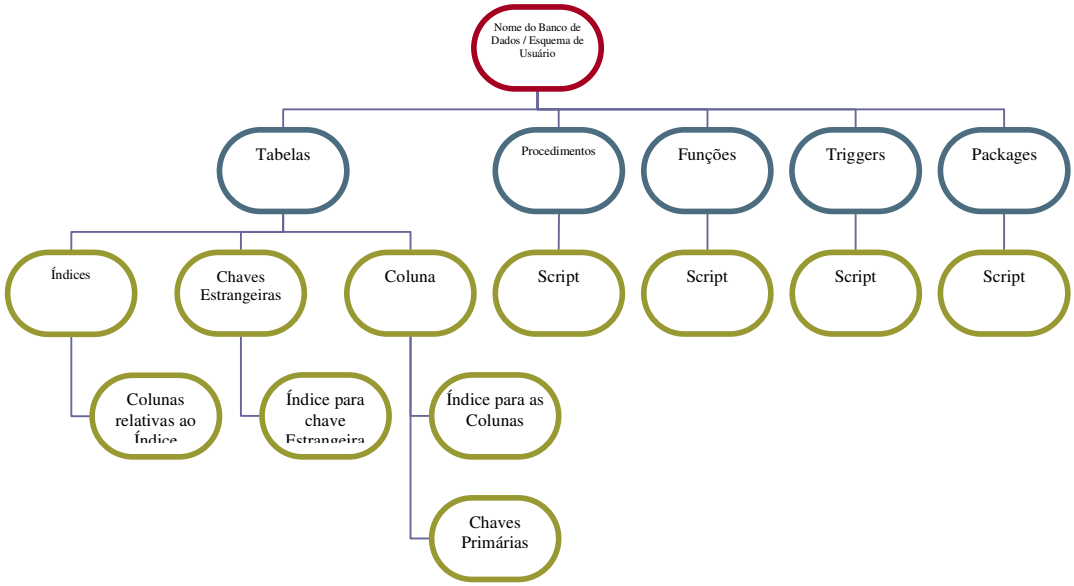

Figura 3.1 - Diagrama da árvore de objetos

Para que a árvore de objetos funcione são gerados os principais nós (tabelas, procedimentos, funções e *triggers*) como é mostrado na Figura 3.2.

Quando o usuário seleciona um desses itens com o mouse, a árvore gera uma ramificação paro o tipo de objeto selecionado como é mostrado na Figura 3.3. Selecionando o nó 'Tabelas' são gerados novos nós para todas as tabelas do banco de dados.

Cada nó representa uma tabela através da imagem de uma tabela e do texto presente no nó.

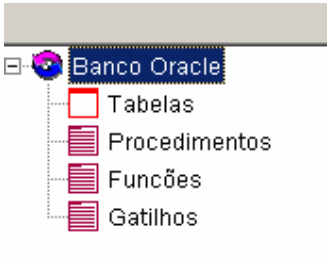

Figura 3.2 - Nós principais da árvore de objetos

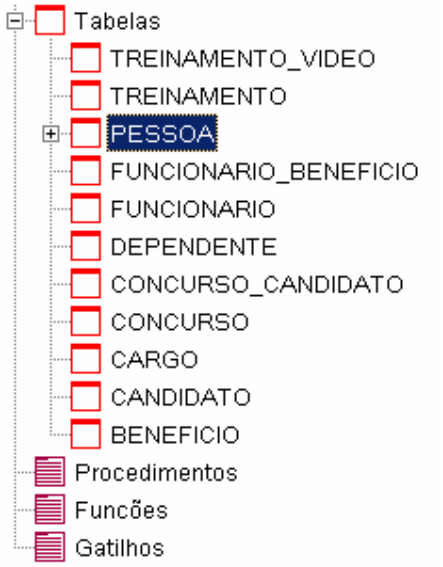

Figura 3.3 - Nós representando tabelas

A *view* necessária para obter-se o nó PESSOA* é USER_TABLES, através do comando SQL "***SELECT table_name FROM USER_TABLES***". O código que implementa essa funcionalidade da árvore esta representado na Figura 3.4.

```
if (ObjectTree.Selected.ImageIndex = IMG_TABLE) then begin

   if not ObjectTree.Selected.HasChildren then begin
      FieldCaption := ObjectTree.Items.AddChildFirst(ObjectTree.Selected, 'Campos');
      FieldCaption.ImageIndex := IMG_FIELD;
      FieldCaption.SelectedIndex := IMG_FIELD;

      USER_TABLE.First;
      USER_TABLE.Locate('TABLE_NAME', ObjectTree.Selected.Text, []);
      repeat

         if not USER_TAB_COLS.Eof then begin
            Field := ObjectTree.Items.AddChild(FieldCaption, USER_TAB_COLSCOLUMN_NAME.AsString);
            Field.ImageIndex := IMG_FIELD;
            Field.SelectedIndex := IMG_FIELD;
            USER_TAB_COLS.Next;
         end;
      until USER_TAB_COLS.Eof;
   end;
end else
```

Figura 3.4 - Trecho de código feito para exibir a árvore de objetos

Novamente, agora selecionando-se o nó 'Tabelas' será criado um nó chamado 'Campos'. Quando se aciona o sinal '+' desse nó aparecem todos os campos da tabela selecionada, conforme pode ser conferido na Figura 3.5..

Para construir os campos de tabelas a partir dos nós que mostram as tabelas, foi usada a tabela USER _TABLES e a tabela USER_TAB_COLS. A *select* utilizada para implementar esta funcionalidade foi: "***SELECT COLUMN_NAME FROM USER_TAB_COLS WHERE USER_TAB_COLS .COLUMN_NAME = USER _TABLES. COLUMN_NAME***".

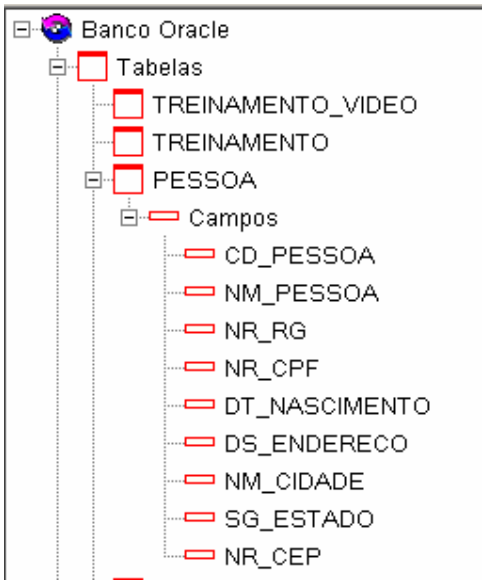

Figura 3.5 - Representação das colunas da tabela 'PESSOA'.

Para visualizar também se um dos campos está definido com chave primária deve-se selecionar o nó. Aparecerá um nó representando o nome do índice de chave primária do campo, como mostrada na Figura 3.6.

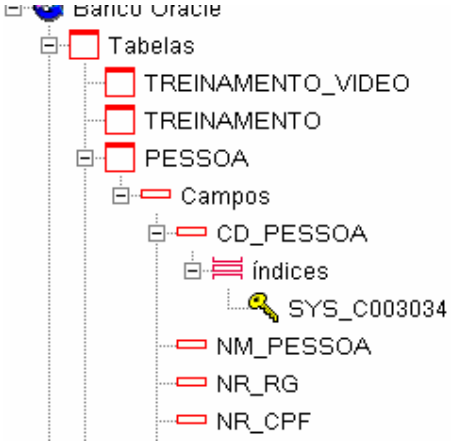

Figura 3.6 - Exibição do índice da chave primária da tabela PESSOA

Se o campo é uma chave primária, é exibida uma pequena chave na cor amarela para o campo.

É possível ainda recuperar o *script* de um procedimento por meio de um menu flutuante que é ativado quando o botão direito do mouse é pressionado, conforme mostrado na Figura 3.7.

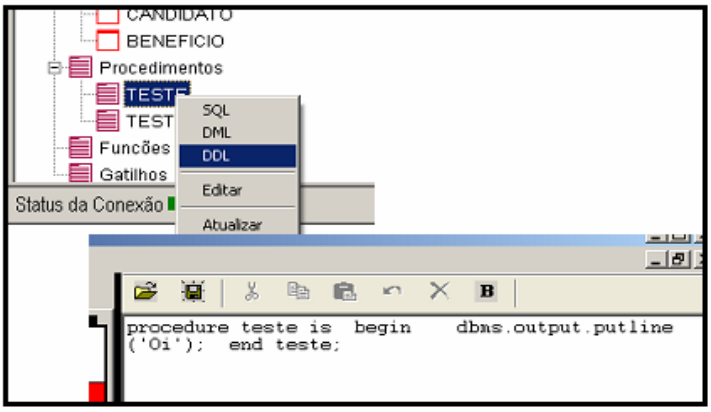

Figura 3.7 - Recuperação de *script* do banco de dados

3.3 Editor/Executor

O editor/executor da ferramenta é chamado a partir de um *thread* (co-processos da aplicação), isto é necessário porque podem ser abertos com finalidades diferentes e para proporcionar a execução concorrente. Para a construção do editor/executor foi usada tecnologia de *threads* e *MDI* para permitir ao usuário a abertura de várias janelas e executar os *scripts* concorrentemente conforme a necessidade do usuário.

Ao se escrever um *script SQL* que execute um cursor realizando operações em uma tabela muito extensa é possível abrir uma, ou mais janelas e executar outros *scripts.*

Para abrir uma nova janela de edição é necessário selecionar a opção arquivo como mostrado na Figura 3.8 e depois selecionar “Novo” como mostrado na Figura 3.9. Na Figura 3.10 mostrada a janela que é criada.

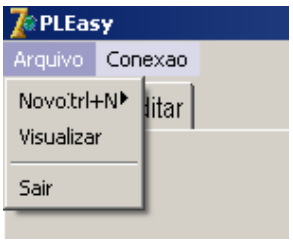

Figura 3.8 – Menu "Arquivo"

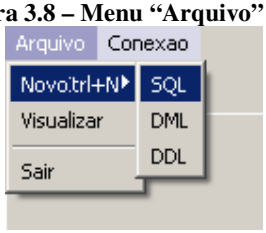

Figura 3.9 – Sub-Menu SQL

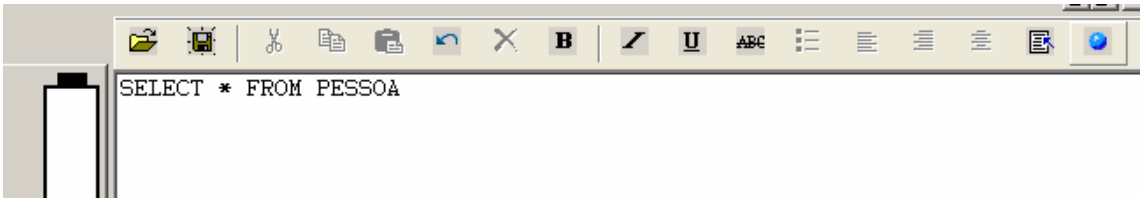

Figura 3.10 - Exemplo de *script* incluído na janela do editor/executor

Para executar o comando *SQL* selecione o botão que aparece selecionado na Figura 3.10 (último à direita), o resultado aparece na Figura 3.11 que mostra as linhas da seleção digitada.

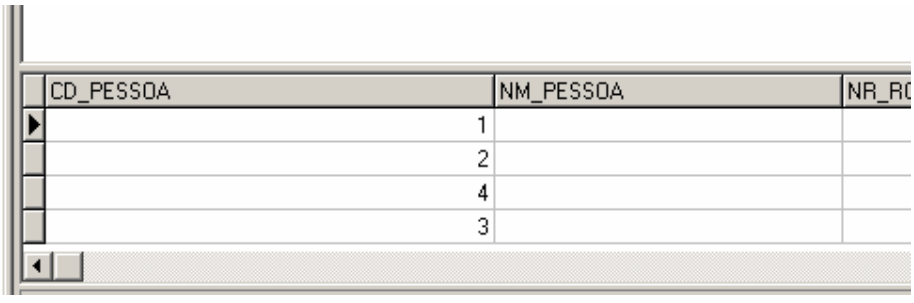

Figura 3.11 - Resultado da execução do *script* da Figura 3.10

É possível também executar seleções mais complexas como seleções de agrupamento, cursores, união entre outras.

3.4 Visualização de Informação – tamanho das tabelas

A visualização proporciona ao usuário um entendimento mais rápido do que está sendo passado pelo programa, isso agiliza a administração do banco de dados e ajuda a entender como as tabelas do banco de dados se comportam.

Foi criada uma nova abstração para representar a relação da quantidade de registros de uma dada tabela e a quantidade de registros de todas as tabelas visíveis pelo usuário do banco de dados: a metáfora de uma bateria.

Para construir a abstração do número de registros de uma tabela por meio de uma bateria foi criado primeiramente um *frame* para conter cada objeto envolvido no processo, esse *frame* contém o desenho de uma bateria (pilha) na cor vermelha, dentro desse desenho foi incluído um retângulo branco representando a quantidade faltante de carga na bateria (para que a carga total da bateria seja representada a altura do retângulo branco deve ser igual a zero) e para alterar a sua altura foi criado um método chamado *Resize*.

Esse *frame* foi incluído no formulário do programa, assim é possível controlar a "carga da bateria" mudando o tamanho da altura do retângulo branco. Na Figura 3.12 é possível visualizar o objeto que representa o retângulo branco, ele aparece selecionado.

O código implementado para alterar o tamanho desse objeto esta na Figura 3.13.

```
var Battery: TBattery;

{ TBatery }

{ TBatery }

procedure TBattery.Resize(Height: Integer);
begin
  Level.Height := Height;
end;

end.
```

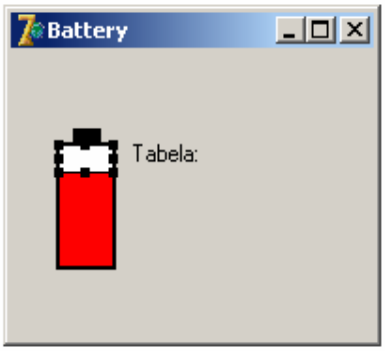

Figura 3.12 – Frame com os objetos que compõem a bateria.

```
procedure TfrmPLEasy.miTamanhoClick(Sender: TObject);
var aux: Integer;
begin
  if (ObjectTree.Selected.ImageIndex = IMG_TABLE) and
     (ObjectTree.Selected.Text <> 'Tabelas') then
  begin
     USER_TABLE.First;
     while not USER_TABLE.Eof do
     begin
        QueryRecCount.Close;
        QueryRecCount.SQL.Text := 'SELECT COUNT(*) "RecNo" FROM ' + USER_TABLETABLE_NAME.AsString;
        QueryRecCount.Active := True;
        aux := aux + QueryRecCount.FieldByName('RecNo').AsInteger;
        USER_TABLE.Next;
     end;

     with Bat do
     begin
        tbEditarDados.Close;
        tbEditarDados.TableName := ObjectTree.Selected.Text;
        tbEditarDados.Open;
        Resize(63 - (tbEditarDados.RecordCount * 63) div aux);
     end;
  end;
end;
```

Figura 3.13 – Trecho do programa que altera a carga da bateria.

Para saber em quantas partes dividir a altura da carga para cada registro da tabela relacionada em relação a todos os registros das tabelas do usuário é feito o cálculo representado na Figura 3.13.

```
procedure TfrmPLEasy.miTamanhoClick(Sender: TObject);
var i, j: Integer;
begin
  j := 0;
  for i := 1 to USER_TABLE.RecordCount do
  begin
    j := j + USER_TAB_COLS.RecordCount;
  end;
  if (ObjectTree.Selected.ImageIndex = IMG_TABLE) and (ObjectTree.Selected.Text <> 'Tabelas') then
  begin
    gdTableGridDblClick(Sender);
    with Bat do
    begin
      tbEditarDados.Close;
      tbEditarDados.Open;
      Resize(63 - (tbEditarDados.RecordCount * 63) div j);
    end;
  end;
end;
```

Figura 3.13 - Trecho de código para calcular a visualização

Analisando a Figura 3.14 pode-se ver que a variação se dá pela modificação da altura do objeto que representa o retângulo.

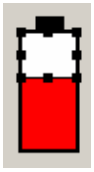

Figura 3.14 - Variação da carga da bateria

3.5 Visualização de Informação – tamanho do BD

Uma outra opção de visualização de informação disponibilizada pela ferramenta é um comparativo geral das quantidades de registros das tabelas por meio de um gráfico de barras.

Nesse gráfico, cada barra representa o número de registros de cada tabela do banco de dados. Na Figura 3.15 é mostrado o gráfico usando um banco de dados de exemplo e na Figura 3.16 é mostrado o trecho de código que gera as barras.

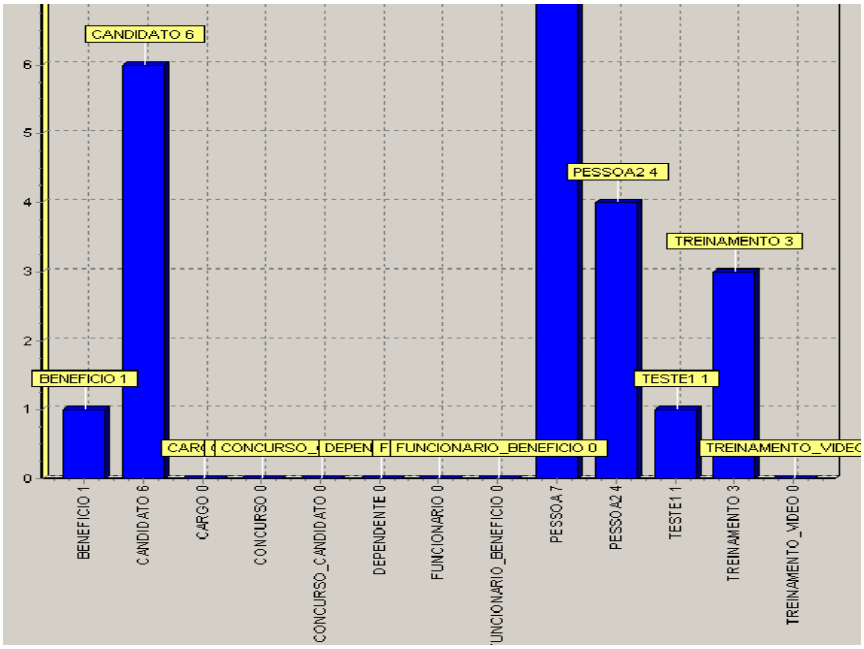

Figura 3.15 - Gráfico em barras do número de registros das tabelas do banco de dados.

```
USER_TABLE.First;
while not USER_TABLE.Eof do
begin
   QueryRecCount.Close;
   QueryRecCount.SQL.Text := 'SELECT COUNT(*) "RecNo" FROM ' + USER_TABLETABLE_NAME.AsString;
   QueryRecCount.Active := True;

   Series1.Add(QueryRecCount.FieldByName('RecNo').Value, USER_TABLETABLE_NAME.AsString + ' ' +
                                        QueryRecCount.FieldByName('RecNo').AsString, clBlue);
   USER_TABLE.Next;
end;
```

Figura 3.16 - Trecho de código que gera as barras

4 Estudo de casos

A ferramenta PLEasy, desenvolvida neste trabalho está apta á reconhecer objetos do banco através de uma árvore de dados, editar e executar scripts DDL, DML e de consulta (*selects*). É capaz também de editar os dados de uma tabela selecionada e mostrar através da metafora de uma bateria a quantidade de registros dessa tabela.

Foram realizados dois estudos de casos com a ferramenta usando-se dois bancos de dados diferentes e avaliando: árvore de dados, o editor/executor de *scrips*, a edição dos dados de uma tabela e, por fim, a metáfora da bateria.

O capítulo foi dividido em 3 tópicos: Árvore de Objetos, Editor/Executor e Metáfora da bateria.

4.1 Árvore de Objetos

Para testar a árvore de objetos foram observadas tabelas de bases diferentes, nas Figuras 4.1 e 4.2 são mostradas as duas árvores de objetos:

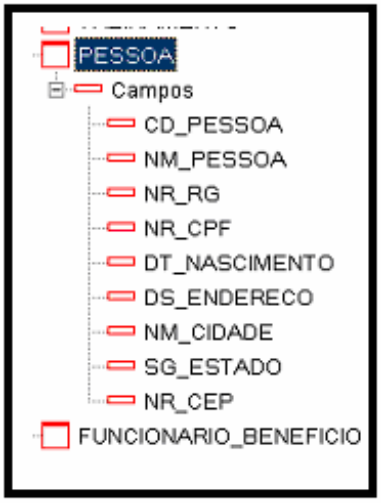

Figura 4.1 – Árvore de objetos para PESSOA do usuário banco de dados *ORCL*

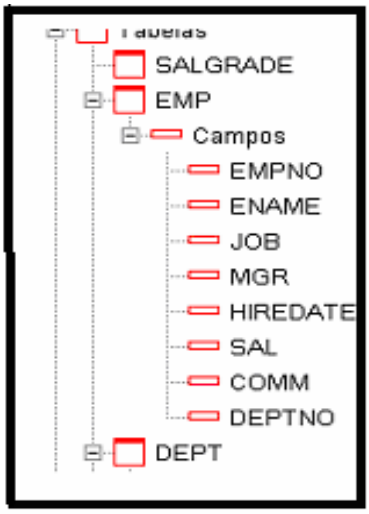

Figura 4.2 – Arvore de objetos para a tabela EMP do banco de dados *Scott*

No banco *ORCL* apareceram todos os campos como era esperado na tabela PESSOA representada pela Figura 4.1, isso também pode ser visualizado na tabela EMP do banco de dados *scott*.

A árvore de objeto reproduz os elementos do SGBD como uma hierarquia de elementos, é costume usar a hierarquia para organizar empresas, família, prioridades, e em muitas outras situações.

4.2 Editor/Executor

O primeiro teste com o editor/executor deste programa foi feito na base de exemplo do banco de dados no usuário *scott*. Para realizar o teste foi criada uma tabela baseada na estrutura de dados da tabela *BONUS* de acordo com a Figura 4.3.

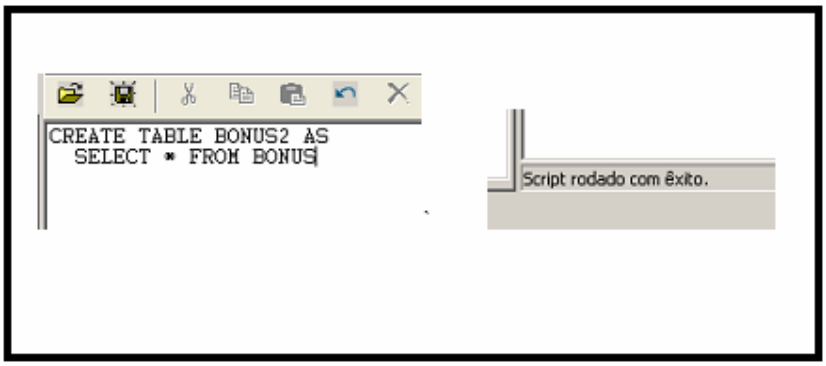

Figura 4.3 - *Script* executado no banco de dados exemplo.

Para verificar se a ferramenta criou a tabela BONUS2 foi usado o programa *SQL *Plus Worksheet* disponível pelo SGBD *Oracle*. Com a instrução "**SELECT * FROM TABS"** é possível exibir as tabelas do banco de dados como mostrado na Figura 4.4 e as tabelas aparecem na Figura 4.5..

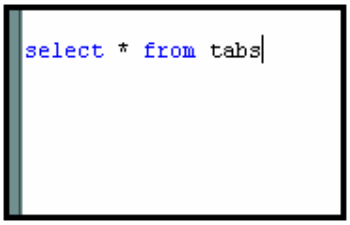

Figura 4.4 – Pesquisando tabelas do usuário scoot

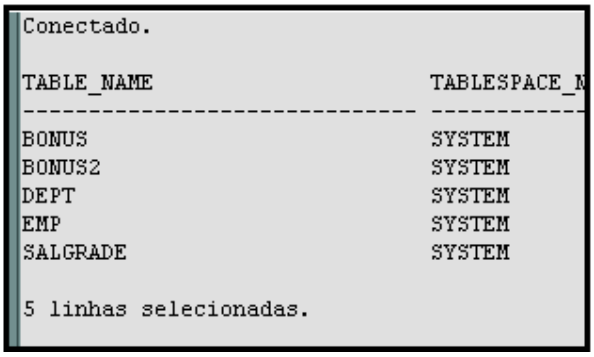

Figura 4.5 – Resultado da pesquisa na base scoot

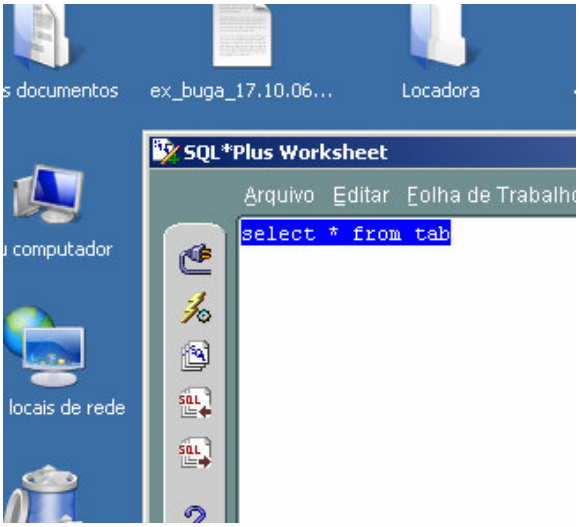

Figura 4.6 – Seleção de todas as tabelas do banco *SQL *Plus Worksheet*

O segundo teste foi feito no banco de dados *ORCL* e foi criada uma tabela baseada na estrutura de dados da tabela PESSOA* e o resultado. Essa operação é mostrada na Figura 4.6.

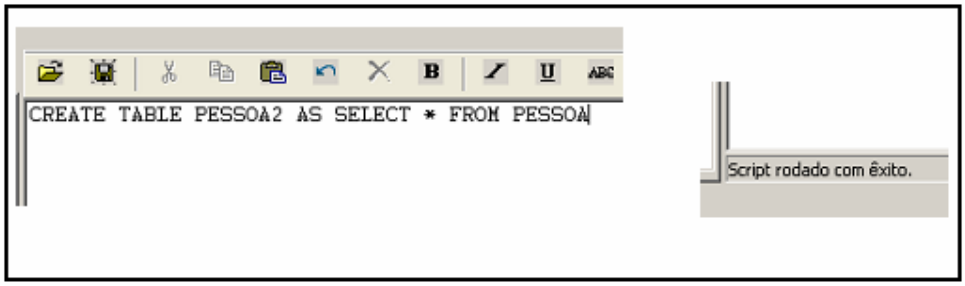

Figura 4.7 - *Script* executado no banco de dados exemplo.

Novamente, para verificar se a ferramenta criou a tabela PESSOA2 foi usado o programa *SQL *Plus Worksheet* disponível pelo SGBD *Oracle* como pode ser visto na Figura 4.8.

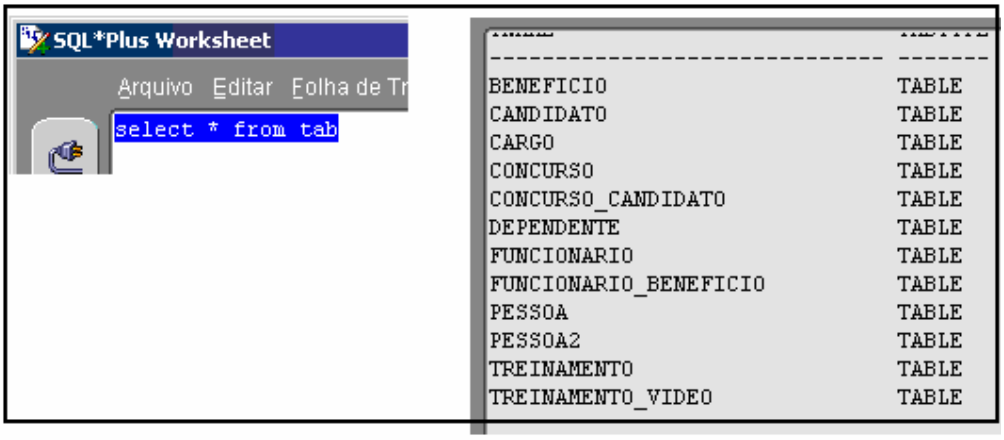

Figura 4.8 - Seleção das tabelas do banco de dados *ORCL*.

4.3 Metáfora da bateria

Para testar a metáfora da bateria no banco de dados *scott* foi usada a tabela DEPT como é observado na Figura 4.9 Nessa Figura pode ser observada a representação da quantidade de registros na tabela em relação ao desenho da carga da bateria.

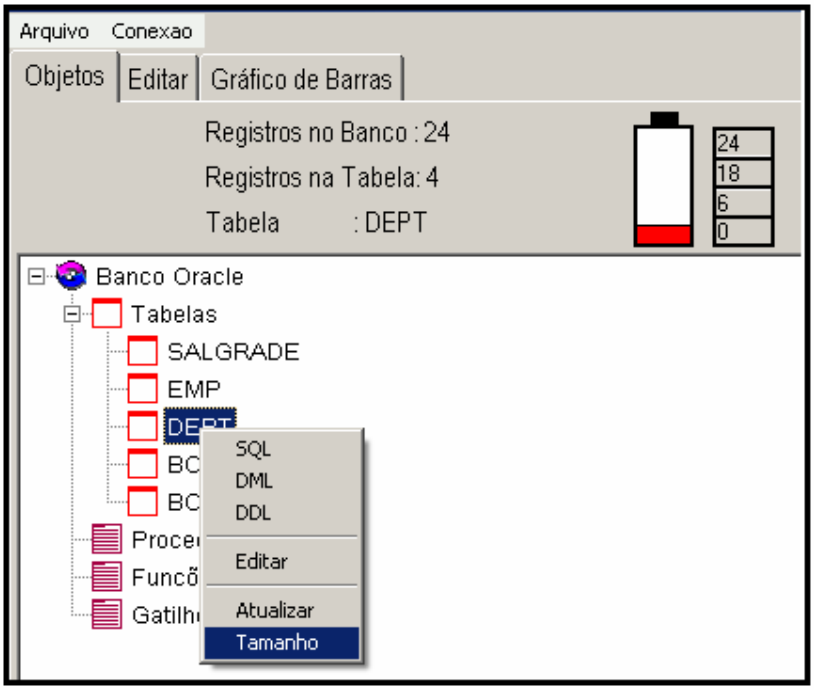

Figura 4.9 - Teste da metáfora da tabela DEPT do banco de dados de exemplo do *Oracle*

O teste da metáfora também foi feito com a tabela EMP do usuário *scoot* do banco de dados de exemplo do *Oracle*, a Figura 4.10.

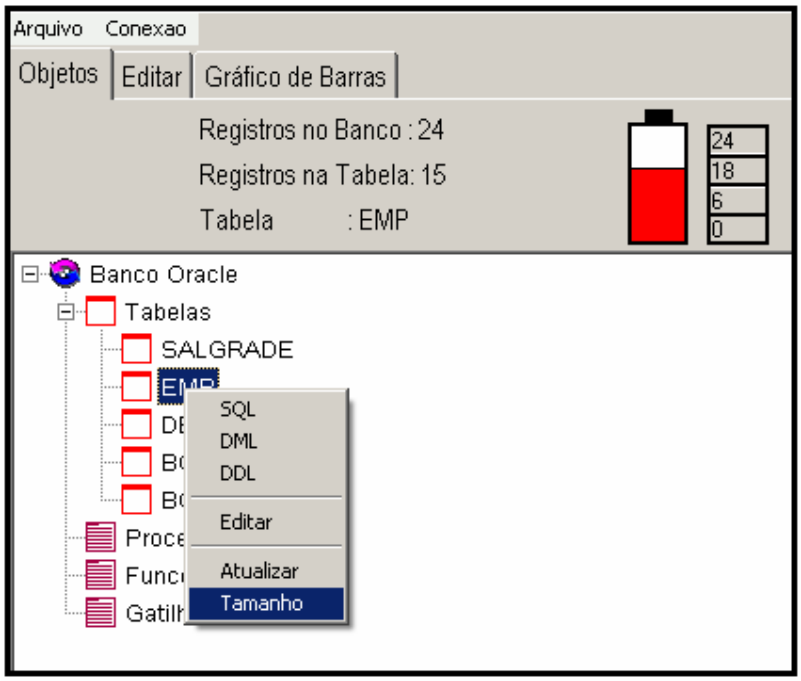

Figura 4.10 – Teste da metáfora da tabela EMP do banco de dados scott

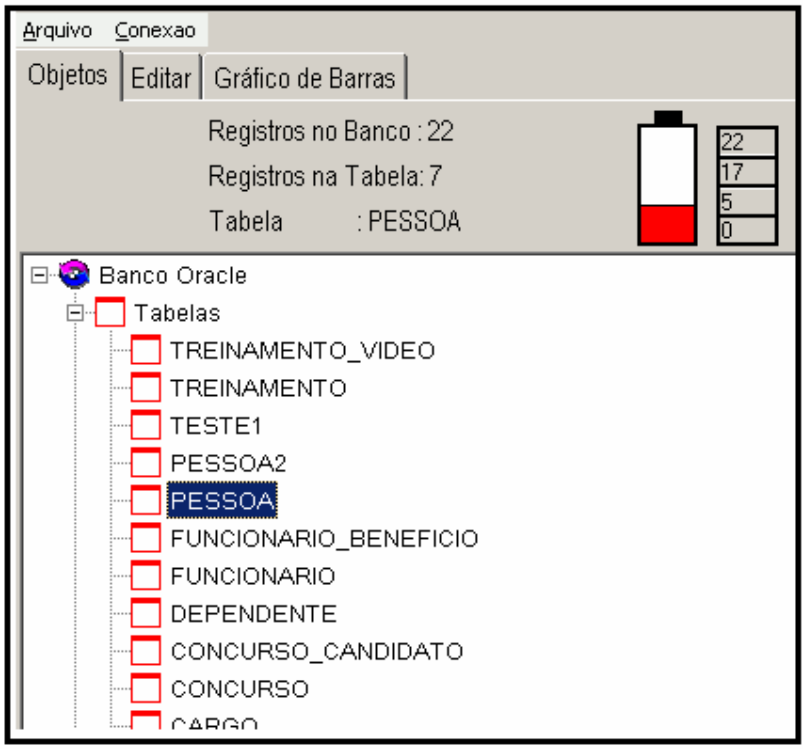

Figura 4.11 – Teste da metáfora da tabela PESSOA do banco de dados *ORCL*

Para testar a metáfora da bateria ainda foi feito um teste em uma tabela do usuário *orcl* do banco de dados *ORCL.* Na Figura 4.11 é mostrado o teste.

Conclusões

O crescimento do uso de SGBD(s) tornou necessário o uso de técnicas de visualização para proporcionar uma compreensão mais rápida e eficaz. A Ferramenta *PLEasy* possui algumas dessas técnicas implementadas.

Como mostra o estudo de caso apresentado neste trabalho, a árvore de objetos tornou-se muito útil para visualizar a estrutura do banco de dados. Ela foi usada para reconhecer os dados usados durante esses testes.

A idéia de desenhar a imagem de uma bateria e abstrair a quantidade de registros de uma dada tabela por meio da visualização de sua carga, foi atingido um diferencial de outras ferramentas disponíveis no mercado.

O editor/executor atingiu as expectativas na execução de comandos DDL, DML e em pesquisas SQL.

Os testes de criação, manipulação e seleção de objetos obtiveram os resultados esperados. Isso faz da ferramenta uma opção viável para o auxílio a programadores e administradores de bancos de dados.

Algumas sugestões para trabalhos futuros são completar a árvore de objetos permitindo a visualização de outros objetos do SGBD como, por exemplo, chaves estrangeiras, *constraints* e packages. Outra idéia é aprimorar o módulo Editor/Executor da ferramenta incluindo a criação de *templates* e o reconhecimento da sintaxe *SQL* do *Oracle* nos *scripts* e avisar sobre erros antes de sua execução. Também é possível gerar outras metáforas para visualização de dados para o SGBD.

Referências

ASADI, M. **Oracle 7.3** Indianapolis: Sams Publishing, 1997.

CARD, S. K.; MACKINLAY, J. D.; SHNEIDERMAN, B. **Information Visualization: Using Vision to Think**. San Diego: Academic Press, 1999.

CANTÚ, M. **Dominando o Delphi 5 – A Bíblia.** São Paulo: Makron Books, 2000

CHEN, C.. **Information Visualization and Virtual Environments**. Singapore: Springer, 1999.

FREITAS, C. M. D. S, CHUBACHI, O. M, Luzzardi, P.R.G.; CAVAL R.A.; **Introdução à Visualização de Informações.** In: Revista Rita, v.8, n.2, p1-16, 2001. Disponível em: <http://www.inf.ufrgs.br/~revista/docs/rita08/rita_v8_n2_p143a158.pdf>. Acesso em 18 nov. 2006.

HARRINGTON, J. L. **Projetos de bancos de dados relacionais : teoria e prática.** Sl.**,** 2002.

KORTH, Henry F.; SILBERSCHATZ, Abraham; SUDARSHAN, S.. **Sistema de banco de dados**. 3ª ed. São Paulo: Makron Books, 1999. 778p.

MACHADO, F. N. R. **Bancos de Dados: Projeto e Implementação.** São Paulo: Ed. Erica 2004

NETO, M. C. M., Mendonça, M., SANTOS C. A. S.; **GraphMiner: Uma Ferramenta de Mineração Visual de Dados em Bases Relacionais.** 2006. In: Webmedia & LA Web, 2004. Disponível em: <http://is.ls.fi.upm.es/jiisic04/Papers/72.pdf>. Acesso em 16 nov. 2006.

ORACLE**; Sobre o Oracle.** Disponível em:

<http://www.oracle.com/global/br/corporate/index.html>. Acesso em: 28 nov. 2006.

TRAINA, A. J. M., TRAINA, C. J.; BOTELHO, E.; BARIONE, M. C. N.; BUENO, R. **Visualização de dados em Sistemas de Bases de Dados Relacionais.** In: XVI Simpósio Brasileiro de Banco de Dados. Anais. Rio de Janeiro/RJ: COPPE/UFRJ, outubro de 2001. p. 95-109.

WIKKIPEDIA; **Oracle.** Disponível em:

<http://pt.wikipedia.org/wiki/Oracle>. Acesso em 28 nov. 2006.

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

RICARDO FILIPE GUIMARÃES GONÇALVES PAULINO

PLEASY: FERRAMENTA DE APOIO AO DESENVOLVEDOR PARA SGBD ORACLE

MARÍLIA
2006

RICARDO FILIPE GUIMARÃES GONÇALVES PAULINO

PLEASY: FERRAMENTA DE APOIO AO DESENVOLVEDOR PARA SGBD ORACLE

Monografia de Trabalho de Conclusão do Curso de Ciência da Computação do Centro Universitário Eurípides de Marília, mantido pela Fundação de Ensino Eurípides Soares da Rocha, para obtenção do Título de Bacharel em Ciência da Computação.

Orientadora:
Prof. Dra. Fátima L. S. Nunes Marques

MARÍLIA

2006

Dedico este trabalho a todos os que me apoiaram na sua execução.

AGRADECIMENTOS

Agradeço o apoio da esposa e família, pelos dias nos quais não pude atendê-los como gostaria de fazê-lo. Agradeço a oportunidade de realizar este trabalho à professora Fátima e aos professores do curso de Ciência da Computação.

Epígrafe

"Não importa quão difícil tenha sido o teu passado, poderás sempre recomeçar hoje."

Buda

PAULINO, Ricardo Filipe Guimarães Gonçalves – **PLEASY**: FERRAMENTA DE APOIO AO DESENVOLVEDOR DE SCRIPTS PARA ORACLE. 2006. 49 F. Trabalho de Conclusão do Curso de Ciência da Computação – Monografia - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2006.

RESUMO

O sistema PLEasy é uma ferramenta de apoio ao desenvolvimento e vizualização para usuários do SGBD Oracle. É um sistema que visa a propiciar algumas facilidades ao administrador de banco de dados, fornecendo um ambiente integrado de desenvolvimento com três recursos: árvore de objetos do banco com tabelas, campos, índices, gatilhos, procedimentos armazenados e funções; editor de textos com funcionalidades como copiar, colar, recortar, alterar a fonte e outros; executor de *scripts* que trabalhe com processamento concorrente utilizando *threads*. É utilizado o dicionário de dados do SGBD Oracle para extrair as informações dos objetos, permitindo ao usuário ver o conteúdo da definição das estruturas de tabelas, procedimentos, funções, gatilhos e outros objetos do banco.

Palavras Chave: Dicionário de Dados, Meta dados, Ferramentas de geração de scripts, Oracle, PLEASY.

PAULINO, Ricardo Filipe Guimarães Gonçalves – PLEASY: FERRAMENTA DE APOIO AO DESENVOLVEDOR DE SCRIPTS PARA ORACLE. 2006. 49 F. Trabalho de Conclusão do Curso de Ciência da Computação – Monografia - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2006.

ABSTRACT

PLEASY is a tool for development support of scripts and includes visualizations for the Oracle DBMS user. It is a system that intend to provide some facilities to the database administrator, bringing him an integrated environment within with three resources: a database object tree with tables, indexes, fields, triggers, stored procedures and functions; a text editor with some functionalities in it like: copy, paste, cut, edit the font type and others; script executer that can work with competitive tasks using *threads*. The tool uses the metadata database to extract the objects information, providing to the user a way to see the structure definitions of tables, procedures, functions, triggers and other database objects..

Keywords: Data dictionary, Meta data, Scripts generating tools, Oracle, PLEasy

ÍNDICE

ÍNDICE DE FIGURAS

LISTA DE ABREVIATURAS

ADO – *ActiveX Data Object*

API - *Application Programming Interface*

DDL - *Data Definition Language*

DLL - *Dynamically Linked Library*

DML - *Data Definition Language*

MDI – *Multi Document Interface*

RAD – *Rapid Application Development*

SGA - *Shared Global Area*

SGBD - *Sistema Gerenciador de Bancos de Dados*

SQL - *Structured Query Language*

INTRODUÇÃO

Uma das dificuldades existentes no desenvolvimento em bancos de dados é a visualização dos objetos envolvidos na criação e manipulação de dados. Isso torna lenta a construção de novos objetos, pois o usuário precisa conhecer a estrutura do banco de dados, que pode ser muito complexa em alguns casos.

Visualização é o termo utilizado para se referir aos mecanismos que permitem a extração de informações de conjuntos complexos de dados. É comum utilizar-se técnicas de computação gráfica, processamento de imagem e realidade virtual para implementar visualizações de dados (BERTI, 2004).

Por meio da visualização é possível utilizar as tecnologias de processamento disponíveis nos computadores mais modernos para oferecer uma percepção rápida de conjuntos de dados (BERTI, 2004).

A visualização usa metáforas como abstração da informação. A escolha de uma metáfora de visualização adequada ao SGBD é fundamental para facilitar o entendimento sobre os dados.

Objetivos do trabalho

Este trabalho representa através de técnicas de visualização da informação estruturas, dados e outros objetos do SGBD de modo a auxiliar programadores no reconhecimento de conjuntos de dados e estruturas do SGBD *Oracle*. Para tanto, foram realizadas pesquisas sobre visualização de dados, sobre o dicionário de dados do SGBD *Oracle* e a construção de objetos gráficos no ambiente *RAD Delphi*.

A ferramenta PLEASY construída neste trabalho apresenta mecanismos de recuperação de informações com o fim de tornar clara a análise de estruturas e dados usando técnicas de visualização.

Disposição do trabalho

Além desta introdução, esta monografia contém os seguintes capítulos:

Capítulo 2 – Comenta sobre as formas de interagir Sistemas Gerenciadores e Visualização, Sistemas de Arquivos, Sistemas Gerenciadores de Bancos de Dados, Modelo Relacional, Visualização da Informação, Dicionário de Dados do SGBD *Oracle* e Trabalhos Correlatos.

- Capítulo 3 – Apresenta a implementação da ferramenta *PLEasy;*
- Capítulo 4 – Apresenta dois estudos de caso com a finalidade de mostrar as funcionalidades da ferramenta.;

Por fim, o último Capítulo – Apresenta as conclusões e, ao final, são disponibilizadas as referências bibliográficas utilizadas para fornecer o embasamento teórico da monografia.

2 SGBD(s) e Visualização de Informação

Existem várias maneiras de se representar dados. Com a visualização vários áreas no desenvolvimento de softwares se beneficiaram. Entre os vários tipos de visualizações podem ser destacadas a científica, a volumétrica, a médica, a geográfica e a de informações abstratas (BERTI, 2004).

"Visualizações baseadas em mapas geográficos são simples, intuitivas e até naturais. Elas quebram a barreira entre um sistema complexo e o conhecimento de uma área de domínio" (CHEN, 1999).

De fato, as informações organizadas sobre estruturas geográficas são bastante intuitivas o que as torna mais compreensíveis, porém isto não ocorre em todo tipo de representação, Chen (1999) relata questões relacionadas à dificuldade de compreensão de dados abstratos comparando-os a visualização geográfica. Como não existe um padrão específico a ser adotado para representarem dados através de uma visualização, ocorrem, em conseqüência disso, certas dificuldades durante a escolha de uma representação adequada (BERTI, 2004).

Com o avanço da tecnologia de armazenamento, bancos de dados tornaram-se maiores e mais complexos, imagens, dados e objetos de diferentes mídias têm sido guardados e recuperados usando uma nova tecnologia: os sistemas gerenciadores de banco de dados, também chamados de SGBD(s) (BERTI, 2004).

A sobrecarga de informações é uma das principais preocupações na representação de resultados obtidos por meio de mecanismos de recuperação de informações. Uma abordagem

para contornar as dificuldades de selecionar as informações relevantes dentre o resultado de buscas é utilizar técnicas de visualização SGBD(s) (BERTI, 2004).

Para que o volume e a complexidade tão grande dos SGBD(s) possam ser assimilados pelo ser humano é necessária a utilização de mecanismos visuais para representar conjuntos de dados por meio de abstrações, ou seja, metáforas (BERTI, 2004).

Por meio dessas técnicas o usuário pode obter uma representação visual, que, se por um lado abstrai detalhes do conjunto de informações, por outro, proporciona a organização desse conjunto segundo algum critério.

2.1 Sistemas de arquivos

Segundo Harrington (2002) os primeiros sistemas de processamento de arquivo eram compostos por um conjunto de arquivos de dados, mais comumente arquivos de texto.

As aplicações que usassem sistemas de arquivos deviam realizar todas as operações necessárias tais como inserção, exclusão, organização das tabelas de registros em diferentes ordens como crescente e decrescente. Se fosse necessário, por exemplo, filtrar dados, isso era feito manualmente.

Harrington (2002), discutindo sobre arquivos de dados, diz que esse tipo de arquivo é organizado em um formato fixo e muito preciso.

De acordo com Harrington (2002) os sistemas realizados com esse modelo de armazenamento em arquivos de dados estão sujeitos alguns problemas, por exemplo:

- A alteração do *layout* de um arquivo de dados (por exemplo, a alteração do tamanho de um campo) requer a alteração de todos os programas que fazem acesso a esse arquivo como também a regravação de todos os dados desse arquivo, pois uma eventual pesquisa para um registro inserido antes da alteração do *layout* do arquivo, resultaria em informações equivocadas (HARRINGTON, 2002).

- A pesquisa de dados específicos neste tipo de estrutura pode se tornar demasiadamente lenta visto que a pesquisa seria seqüencial e deveria percorrer todos os bytes até chegar-se à informação desejada (HARRINGTON, 2002).

- A principal vantagem de um sistema de processamento de arquivos é que ele é simples e mais barato, porém, um sistema deste tipo fica sujeito a inconsistências como, por exemplo, a redundância desnecessária de dados que provoca, inevitavelmente incongruência de dados (HARRINGTON, 2002).

Devido a esses problemas surgiu a necessidade de desenvolver softwares que facilitassem a tarefa de manipulação de dados, denominados Sistemas Gerenciadores de Bancos de Dados (SGBD).

2.2 Sistemas Gerenciadores de Bancos de Dados

Segundo Machado (2004), um banco de dados pode ser definido como um conjunto de dados devidamente relacionados. Pode-se compreender como dados os objetos conhecidos que podem ser relacionados e que possuem significado implícito. Porém o significado do termo banco de dados é mais restrito do que a definição dada anteriormente.

Um banco de dados possui as seguintes características:

- É uma coleção lógica coerente de dados com um significado inerente; uma disposição desordenada dos dados não pode ser referenciada como um banco de dados (MACHADO, 2004).
- É projetado, construído e populado com valores para um propósito específico; um banco de dados possui um conjunto pré-definido de usuários e aplicações (MACHADO, 2004).
- Representa algum aspecto do mundo real, o qual é chamado de minimundo, qualquer alteração efetuada no minimundo é automaticamente refletida no banco de dados (MACHADO, 2004).

Conforme afirmam Korth, Silberchatz e Sudarshan (1999) um SGBD é constituído por um conjunto de dados associados a um conjunto de programas para acesso a esses dados.

A intenção no desenvolvimento de banco de dados é gerir um negócio de maneira mais eficiente, rapidamente e com segurança. Em SGBD(s), dados são inter-relacionados para garantir integridade e eficiência na realização de pesquisas. O SGBD fornece estruturas, regras e funções para facilitar essa implementação (MACHADO, 2004).

Na definição de estruturas para os dados usa-se a Linguagem chamada DDL (*Data Definition Language*), para a manipulação de tabelas existe a linguagem DML (*Data Manipulation Language*) (MACHADO, 2004).

2.3 Modelo Relacional

De acordo com Machado (2004) um novo conceito de banco de dados começou a ser usado e implementado nas linguagens no final dos anos 80 com a proposta do modelo relacional.

Esse modelo foi criado no final dos anos 70 por "Edgar F. Codd" e, somente, começou a ser utilizado por um número maior de empresas por volta de 1987. Esse modelo se refere às estruturas de dados como relações matemáticas. Este é um modelo lógico que deu origem à teoria relacional (MACHADO, 2004).

A teoria relacional estuda os conceitos lógicos dos bancos de dados relacionais e nela estabeleceram-se várias regras para descrever a tabela relacional. Em 1979 Edgar F. Codd e Chris Date refinaram o modelo relacional dando origem ao que se chama hoje de modelo relacional estendido (MACHADO, 2004).

A visão utilizada pela teoria relacional é de que o banco de dados é um conjunto de tabelas com linhas e colunas como mostra o exemplo da Figura 2.1 A intenção dessa forma de tratar os dados é fornecer transparência aos usuários, programadores ou não, de forma que não seja necessário que se saiba onde e como estão os dados do sistema. (MACHADO, 2004)

Registro	Código	Nome	Idade	Sexo
1	10	Luis	58	Masculino
2	20	Ricardo	28	Masculino
3	30	Vilma	31	Feminino
4	40	Delfina	56	Feminino

Figura 2.1 - Tabela

Foram estabelecidos critérios para a teoria relacional:

- Cada uma das tabelas é chamada de relação;
- O conjunto de uma linha de colunas é chamado de tupla;
- Cada coluna tem um nome específico e representa um domínio diferente;
- Não há duas linhas iguais;
- Cada tabela tem um nome único no Banco de Dados;
- A ordem das linhas e das colunas não é importante.

Algumas das principais operações para se dar manutenção em um Banco de Dados serão apresentadas a seguir.

Para realizar pesquisas no banco de dados usa–se uma linguagem chamada *SQL.* Machado (2004) lembra que SQL só previa o seu uso de forma interativa. Após sofrer alguns acréscimos, ela passou também a ter capacidade de ser utilizada em linguagens hospedeiras, tais como *Cobol, Fortran*, *"C"*.

Machado (2004) afirma que a Linguagem *SQL* assume um papel muito importante nos SGBD(s), podendo ter muitos enfoques:

- **Linguagem Interativa de Consulta** (*Query AdHoc*) – com os comandos *SQL* os usuários podem montar consultas poderosas sem a necessidade de criar um programa podendo utilizar *forms* ou ferramentas de montagem de relatório (MACHADO, 2004).

- **Linguagem de Programação para Acesso a Banco de Dados** – comandos *SQL* embutidos em programas de aplicação que acessam os dados armazenados (MACHADO, 2004).

- **Linguagem de Administração de Bancos de Dados** – responsável pela administração do banco de dados (*Data Base Administrator*) pode utilizar comandos *SQL* para realizar suas tarefas (MACHADO, 2004).

- **Linguagem Cliente/Servidor** – os programas (Cliente) dos computadores pessoais usam comandos *SQL* para se comunicarem por meio de uma rede local, compartilhando os dados armazenados em um único local (Servidor). A arquitetura cliente/servidor minimiza o tráfego de dados pela rede (MACHADO, 2004).

- **Linguagem para bancos de dados distribuídos** – a *SQL* auxilia na distribuição dos dados através de vários nós conectados ao sistema de aplicação. Auxilia também na comunicação de dados com outros sistemas (MACHADO, 2004).

- **Caminho de acesso a outros bancos de dados em diferentes máquinas** - a *SQL* auxilia na conversão entre diferentes produtos de banco de dados colocados em diversas máquinas (de micro até *mainframes*) (MACHADO, 2004).

- **Definição de dados (DDL)** – permite ao usuário a definição da estrutura e organização dos dados armazenados e as relações entre eles (MACHADO, 2004).

- **Manipulação de dados (DML)** – permite ao usuário ou a um programa de aplicação a inclusão, remoção, seleção ou atualização de dados previamente armazenados no banco (MACHADO, 2004).

- **Controle de acesso** – protege os dados de manipulações não autorizadas (MACHADO, 2004).

- **Compartilhamento de dados** – coordena o compartilhamento dos dados por usuários concorrentes, sem, contudo, interferir na ação de cada um deles (MACHADO, 2004).

- **Integridade dos dados** – auxilia no processo de definição da integridade dos dados, protegendo contra corrupções, inconsistências e falhas do sistema de computação (MACHADO, 2004).

2.4 Visualização da Informação

Desde 1786 a necessidade de se representarem dados utilizando imagens tem sido tema para pesquisas, John Playfair (CARD *et al*., 1999) - geólogo e matemático escocês trabalhava com representação de dados gráficos.

O fundamento da Visualização de Informação está diretamente ligado à cognição humana. A representação é associada com essa peculiaridade do ser humano de modo a facilitar sua memorização e compreensão.

O conceito da Visualização de Informação é o uso de representação visual, interativa e com suporte computacional de dados abstratos para ampliar a cognição humana. Através de uma imagem é possível distinguir com um rápido olhar o que se julgava difícil (CARD *et al*., 1999).

2.5 Dicionário de Dados do SGBD Oracle

"O negócio da Oracle é a informação: como gerenciar, usar, compartilhar, proteger. Há quase três décadas a Oracle, maior empresa de software empresarial do mundo, fornece software e serviços que possibilitam às empresas extrair as informações mais precisas e atualizadas de seus sistemas de negócios." (ORACLE, 2006).

O *Oracle* é um SGBD que surgiu no final dos anos 70, quando Larry Ellison vislumbrou uma oportunidade que outras companhias não haviam percebido, quando encontrou uma descrição de um protótipo funcional de um banco de dados relacional e descobriu que nenhuma empresa tinha se empenhado em comercializar essa tecnologia.(WIKIPEDIA, 2006).

Ellison e os co-fundadores da *Oracle Corporation*, Bob Miner e Ed Oates, perceberam que havia um tremendo potencial de negócios no modelo de banco de dados relacional tornando assim a maior empresa de software empresarial do mundo (WIKIPEDIA, 2006).

O SGBD da *Oracle* é líder de mercado. O *Oracle* 9i foi pioneiro no suporte ao modelo *web*. O *Oracle* 10g, mais recente, se baseia na tecnologia de *grid.* Além da base de dados, a Oracle desenvolve uma *suíte* de desenvolvimento chamada de *Oracle Developer Suite*, utilizada na construção de programas de computador que interagem com a sua base de dados. Além disso, a *Oracle* também criou a linguagem de programação PL/SQL, utilizada no processamento de transações (WIKIPEDIA, 2006).

O SGBD Oracle fornece um dicionário de dados, sua função é armazenar as estruturas dos dados. Costuma-se dizer que o "dicionário de dados contém dados sobre os dados do banco", é por meio do dicionário de dados que podemos recuperar informações como, por exemplo, o nome de todas as tabelas contidas em um esquema de usuário ou saber qual o tipo das colunas das tabelas.

O administrador do banco de dados precisa ter conhecimentos sobre como funciona, como recuperar informações úteis que não são visíveis ao usuário comum e até mesmo

reparar danos no BD. Além disso, o dicionário de dados tem um papel importante na manutenção da integridade e segurança do banco de dados.

Ao executar um *script* de procedimento, função, *trigger* ou outros, o interpretador do *Oracle* verifica a sintaxe, os nomes dos objetos, a sua ordem, a seqüência, a escrita dos comandos e a escrita de operadores (ASADI, 1997).

O *Oracle* pode ainda recuperar objetos utilizados nas operações realizadas, mantê-los na área compartilhada do banco de dados (ASADI, 1997).

A principal tabela do dicionário de dados do *Oracle* 9i é a *view DICTIONARY*, que descreve o nome de todas as outras *views* contidas na memória compartilhada do banco. Na Figura 2.2 são mostradas algumas das *views* usadas neste trabalho e sua descrição, no Capítulo 3 são exibidos trechos de código no qual elas são usadas.

Tabelas	Comentário
USER_ALL_TABLES	Descrição de todas os objetos e tabelas relacionais pertencente ao usuário
USER_CONSTRAINTS	Definição de "constraints" nas tabelas pertencentes ao usuário
USER_SOURCE	Conteúdo de objetos armazenados acessíveis ao usuário
USER_TABLES	Descrição das tabelas de usuário
USER_TAB_COLS	Colunas de tabelas, *views* e *clusters*
USER_TAB_COLUMNS	Colunas de tabelas, *views* e *clusters*
USER_TRIGGERS	Triggers (Gatilhos) pertencentes ao usuário
USER_TRIGGER_COLS	Coluna usada nas Triggers (Gatilhos) do usuário
USER_USERS	Informações sobre o usuário corrente
USER_VIEWS	Descrição das views pertencentes ao usuário

Figura 2.2 – Descrição de algumas *views* usadas neste trabalho

2.6 Trabalhos Correlatos

Na literatura são citadas algumas ferramentas construídas para o desenvolvimento de *scripts* em *PL/SQL* e visualização de informação. Entre as ferramentas pesquisadas são apresentados pelo menos um dos seguintes recursos: visualização de objetos do banco, editores de texto para *scripts* e *recuperação de estruturas.*

As ferramentas escolhidas como modelos reconhecem os objetos do SGBD e mostram de alguma forma uma visualização particular do banco de dados e seus objetos.

2.6.1 A Ferramenta FastMapDB

O trabalho de TRAINA *et al.* (2001) apresenta uma ferramenta que disponibiliza de forma gráfica os dados armazenados em uma base relacional.

Essa ferramenta permite que aos usuários participarem do processo de descoberta de conhecimento sobre os dados em análise. O trabalho desenvolve um conceito que suporta a transformação sobre um conjunto de dados permitindo a visualização de atributos de diversos tipos (números, datas e textos). A ferramenta realiza transformações de dados multidimensionais um espaço tridimensional, utilizando uma função de distância definida pelo usuário.

A Ferramenta *FastMap DB* baseia-se na aplicação da técnica *FastMap*, inicialmente desenvolvida com o objetivo de redução de dimensionalidade em espaços de altas dimensões. Na Figura 2.3 é mostrada uma interface da ferramenta

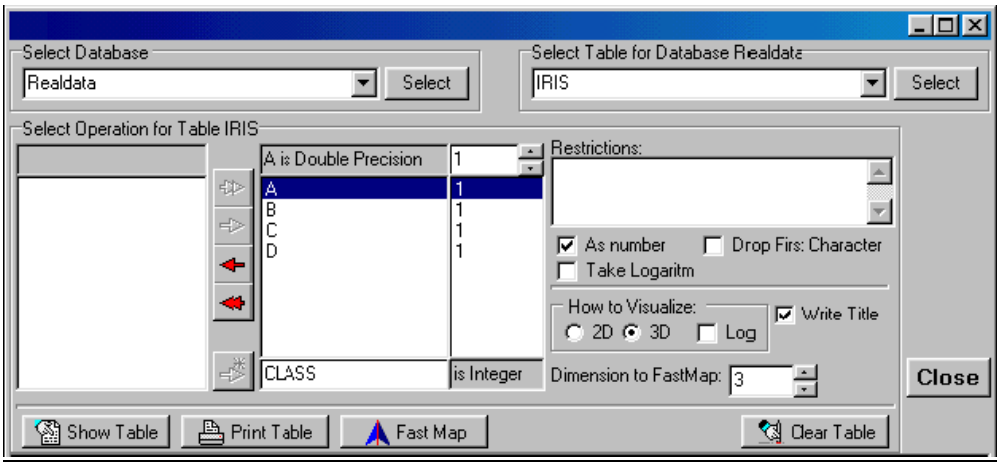

Figura 2.3 - Ferramenta FastMapDB

Na Figura 2.4 é mostrado um experimento detalhado deste trabalho, que visualiza o conjunto dos votos do congresso americano em 1984. Cada registro corresponde ao voto de um congressista em 16 assuntos (por exemplo, gastos com educação, crime, etc.).

Todos os atributos têm o valor 1 (aprova), -1 (não aprova) ou zero (nulo ou abstenção). Cada t-upla tem também um atributo categórico, indicando a que partido, Republicano ou Democrata o congressista correspondente pertence .

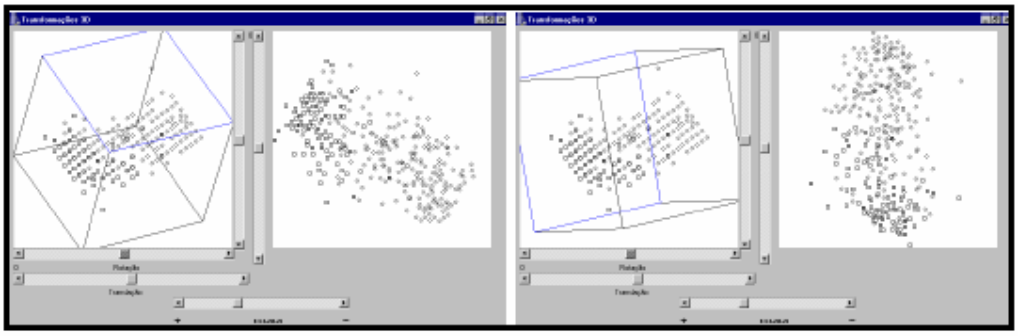

Figura 2.4 – Dois ângulos de visualização do conjunto votos.

2.6.2 GraphMiner: Uma Ferramenta de Mineração Visual de Dados em Bases Relacionais

No trabalho de NETO et al. (2004) foi desenvolvida uma ferramenta chamada *GraphMiner* com o objetivo de apoiar a descoberta de informações úteis em bancos de dados relacionais através da representação gráfica dos objetos do banco na forma de um grafo.

Esse trabalho oferece um mecanismo para a exploração de dados relacionais de forma simples. Além da criação de grafos para representar objetos do banco de dados, esta ferramenta é independente do banco de dados.

Para a construção de um grafo a partir de um banco relacional a ferramenta *GraphMiner* realiza quatro etapas básicas:

1. Escolher uma entidade raiz do banco que se quer visualizar.
2. Descobrir quais são as entidades que se relacionam com a entidade raiz.
3. Selecionar os registros da entidade raiz e de cada uma das entidades relacionadas.
4. Mapear os registros como nós do grafo e as relações entre elas como arestas.

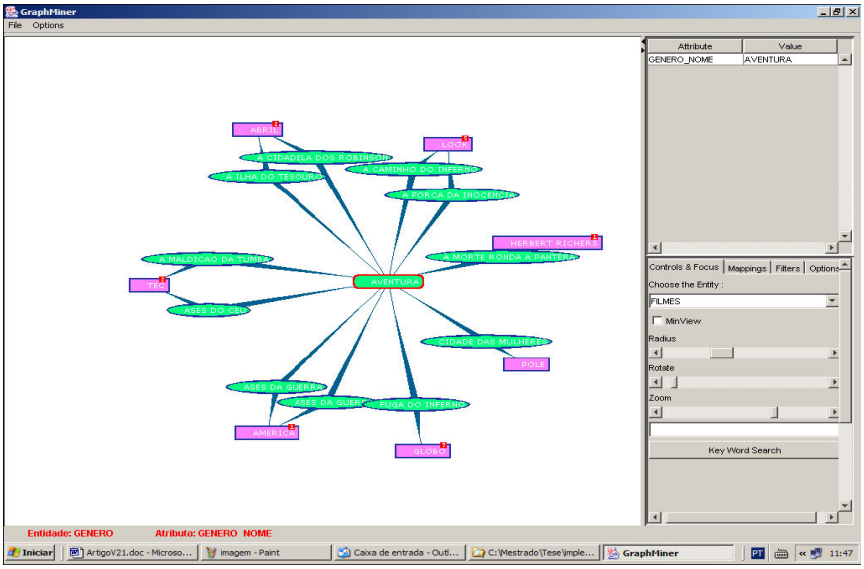

Figura 2.5 - Visualização de dados e suas relações através da *GraphMiner*.

Uma outra funcionalidade que segue a mesma linha do controle e aproximação é o controle do raio de exibição como mostrada na Figura 2.5.

A partir desta funcionalidade a Ferramenta *GraphMine* permite controlar a quantidade de nós exibidos na tela usando como parâmetro para mostrar ou ocultar um nó, a sua distância do nó selecionado. Neste caso, a distância entre dois nós corresponde à quantidade de arestas entre o nó de origem selecionado e nó destino.

A ferramenta disponibiliza ainda os controles de interação que permitem o mapeamento de atributos visuais exibidos no grafo como a cor, o formato e o rótulo dos nós e arestas com qualquer um dos atributos do banco de dados relacional.

2.6.3 VRVis: Ferramenta de Realidade Virtual para Visualização de Informações

O objetivo do trabalho *VRVis* é apresentar uma ferramenta para visualizar grandes volumes de informações abstratas a partir de técnicas de Realidade Virtual com o fim de apresentar a representação da informação de forma intuitiva para o usuário.

Esse trabalho estrutura-se no estudo e utilização de técnicas de Realidade Virtual para Visualização de Informação. Foi usado um banco de dados de imagens de dados médicos com grande volume de informações.

Segundo Berti (2004) grande parte das ferramentas que representam dados de forma gráfica utilizam histogramas ou gráficos de colunas e linhas. Desta forma, uma ferramenta de visualização que utiliza técnicas de Realidade Virtual para demonstrar os dados em consultas pode ter grande contribuição no aspecto da compreensão e percepção das características destes dados.

Na Figura 2.6 é exibida a tela de seleção de parâmetros da consulta da ferramenta de Visualização de Informação VRVis.

Figura 2.6 - Tela de seleção de parâmetros da ferramenta de visualização VRVis.

Por meio da tela apresentada na Figura 2.6, o usuário seleciona os parâmetros para filtragem e composição da consulta.

3 Ferramenta PLEasy

Este capítulo trata da ferramenta construída neste trabalho, versará sobre partes importantes de seu código e funcionamento como classes e métodos. O capítulo também discutirá sobre as tecnologias usadas durante o seu desenvolvimento.

Para se manipular um SGBD é necessário ter conhecimentos sobre suas estruturas e comandos. Em sistemas que usam a linguagem *SQL* como padrão, para criação e manipulação de suas estruturas, o programador precisa estar consciente sobre a versão da linguagem e dos comandos necessários para manipulá-lo. Desta forma, a ferramenta *PLEasy* é um programa implementado em Delphi que fornece funcionalidades e visualizações para SGBD *Oracle,* proporcionando ao programador um ambiente integrado para o desenvolvimento, manipulação de dados e estruturas, e reconhecimento visual de dados através de técnicas de "Visualização da Informação".

Para dar ênfase ao diferencial do programa serão descritas partes de seu código usado para o desenvolvimento da visualização de uma bateria, a árvore de objetos do banco e o Editor/Executor de *scripts*.

3.1 Tecnologias utilizadas

Para implementar o *PLEasy* foram utilizadas diversas tecnologias, entre elas a Ferramenta Delphi, o SGBD *Oracle* e a tecnologia ADO da *MicroSoft.* Foram implementadas basicamente quatro funcionalidades relacionadas à visualização de dados e auxílio ao uso da linguagem SQL: árvore de objetos, editor/executor de comandos, metáfora de bateria e gráfico de barras.

3.2 Árvore de Objetos

A árvore de objetos de banco de dados é uma funcionalidade do programa que mostra inicialmente um nó chamado 'Banco Oracle' de onde derivam os nós 'Tabelas', 'Procedimentos', 'Funções' e 'Gatilhos'. Foi construída usando-se para isto os componentes *TTreeView*, *TImageList* além dos componentes de acesso ao banco de dados. Para se obter os objetos foi usado o dicionário de dados *Oracle*. Na Figura 3.1 é mostrada um diagrama da árvore de dados.

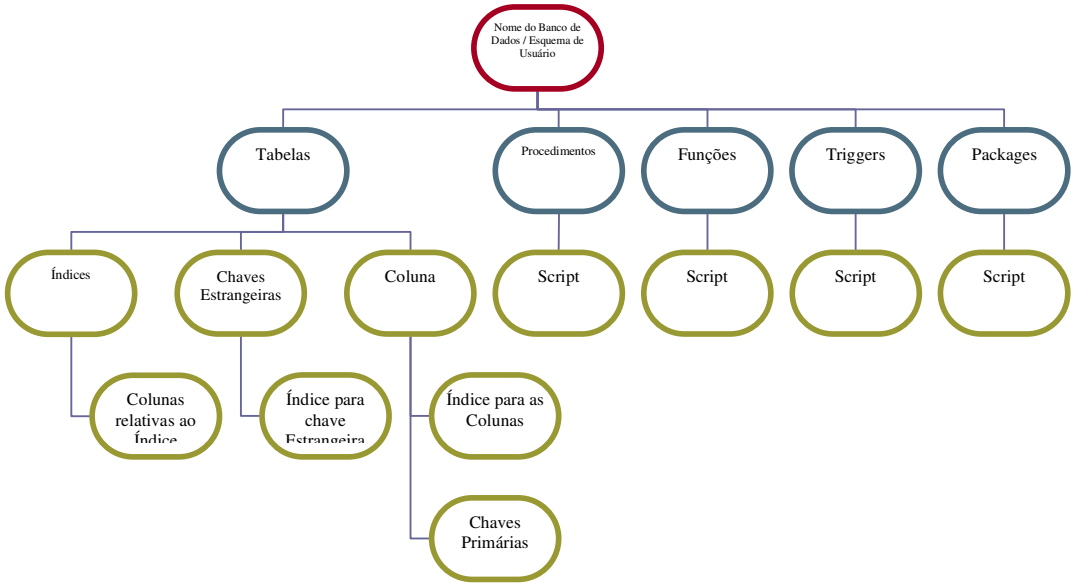

Figura 3.1 - Diagrama da árvore de objetos

Para que a árvore de objetos funcione são gerados os principais nós (tabelas, procedimentos, funções e *triggers*) como é mostrado na Figura 3.2.

Quando o usuário seleciona um desses itens com o mouse, a árvore gera uma ramificação paro o tipo de objeto selecionado como é mostrado na Figura 3.3. Selecionando o nó 'Tabelas' são gerados novos nós para todas as tabelas do banco de dados.

Cada nó representa uma tabela através da imagem de uma tabela e do texto presente no nó.

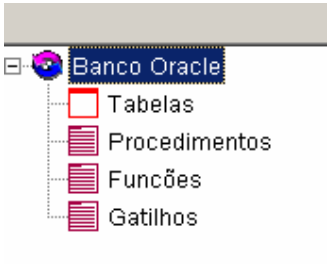

Figura 3.2 - Nós principais da árvore de objetos

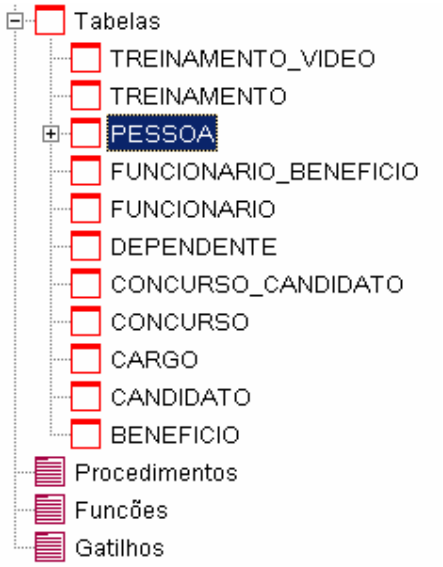

Figura 3.3 - Nós representando tabelas

A *view* necessária para obter-se o nó PESSOA* é USER_TABLES, através do comando SQL "***SELECT table_name FROM USER_TABLES***". O código que implementa essa funcionalidade da árvore esta representado na Figura 3.4.

```
if (ObjectTree.Selected.ImageIndex = IMG_TABLE) then begin

   if not ObjectTree.Selected.HasChildren then begin
      FieldCaption := ObjectTree.Items.AddChildFirst(ObjectTree.Selected, 'Campos');
      FieldCaption.ImageIndex := IMG_FIELD;
      FieldCaption.SelectedIndex := IMG_FIELD;

      USER_TABLE.First;
      USER_TABLE.Locate('TABLE_NAME', ObjectTree.Selected.Text, []);
      repeat

         if not USER_TAB_COLS.Eof then begin
            Field := ObjectTree.Items.AddChild(FieldCaption, USER_TAB_COLSCOLUMN_NAME.AsString);
            Field.ImageIndex := IMG_FIELD;
            Field.SelectedIndex := IMG_FIELD;
            USER_TAB_COLS.Next;
         end;
      until USER_TAB_COLS.Eof;
   end;
end else
```

Figura 3.4 - Trecho de código feito para exibir a árvore de objetos

Novamente, agora selecionando-se o nó 'Tabelas' será criado um nó chamado 'Campos'. Quando se aciona o sinal '+' desse nó aparecem todos os campos da tabela selecionada, conforme pode ser conferido na Figura 3.5..

Para construir os campos de tabelas a partir dos nós que mostram as tabelas, foi usada a tabela USER _TABLES e a tabela USER_TAB_COLS. A *select* utilizada para implementar esta funcionalidade foi: "***SELECT COLUMN_NAME FROM USER_TAB_COLS WHERE USER_TAB_COLS .COLUMN_NAME = USER _TABLES. COLUMN_NAME***".

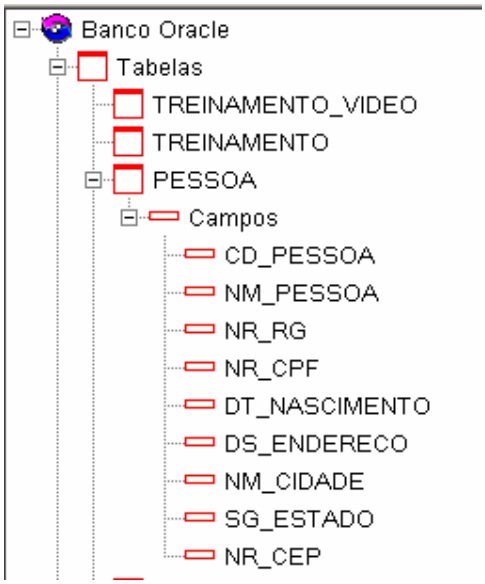

Figura 3.5 - Representação das colunas da tabela 'PESSOA'.

Para visualizar também se um dos campos está definido com chave primária deve-se selecionar o nó. Aparecerá um nó representando o nome do índice de chave primária do campo, como mostrada na Figura 3.6.

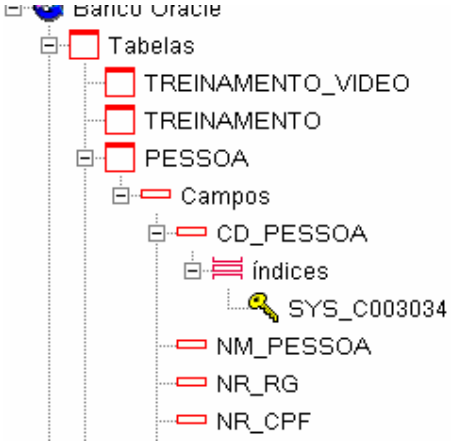

Figura 3.6 - Exibição do índice da chave primária da tabela PESSOA

Se o campo é uma chave primária, é exibida uma pequena chave na cor amarela para o campo.

É possível ainda recuperar o *script* de um procedimento por meio de um menu flutuante que é ativado quando o botão direito do mouse é pressionado, conforme mostrado na Figura 3.7.

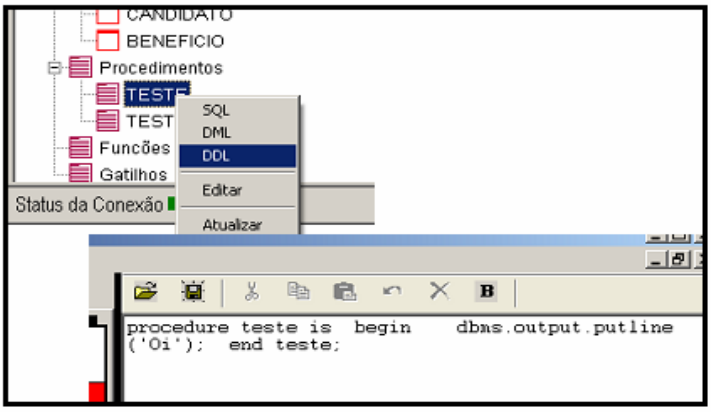

Figura 3.7 - Recuperação de *script* do banco de dados

3.3 Editor/Executor

O editor/executor da ferramenta é chamado a partir de um *thread* (co-processos da aplicação), isto é necessário porque podem ser abertos com finalidades diferentes e para proporcionar a execução concorrente. Para a construção do editor/executor foi usada tecnologia de *threads* e *MDI* para permitir ao usuário a abertura de várias janelas e executar os *scripts* concorrentemente conforme a necessidade do usuário.

Ao se escrever um *script SQL* que execute um cursor realizando operações em uma tabela muito extensa é possível abrir uma, ou mais janelas e executar outros *scripts.*

Para abrir uma nova janela de edição é necessário selecionar a opção arquivo como mostrado na Figura 3.8 e depois selecionar “Novo” como mostrado na Figura 3.9. Na Figura 3.10 mostrada a janela que é criada.

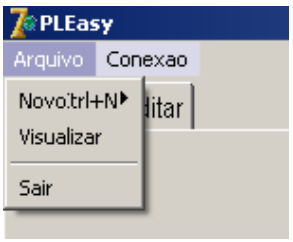

Figura 3.8 – Menu "Arquivo"

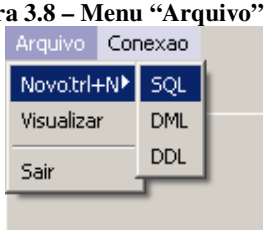

Figura 3.9 – Sub-Menu SQL

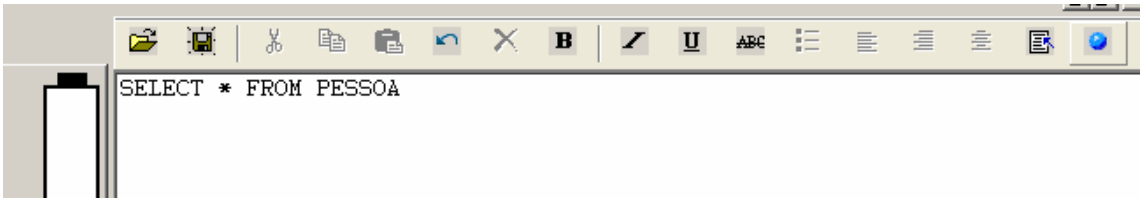

Figura 3.10 - Exemplo de *script* incluído na janela do editor/executor

Para executar o comando *SQL* selecione o botão que aparece selecionado na Figura 3.10 (último à direita), o resultado aparece na Figura 3.11 que mostra as linhas da seleção digitada.

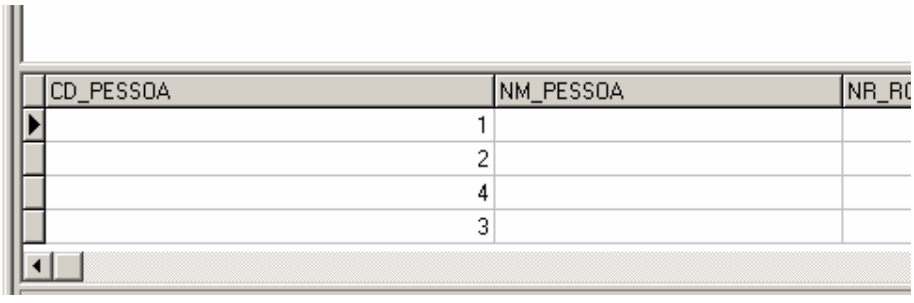

Figura 3.11 - Resultado da execução do *script* da Figura 3.10

É possível também executar seleções mais complexas como seleções de agrupamento, cursores, união entre outras.

3.4 Visualização de Informação – tamanho das tabelas

A visualização proporciona ao usuário um entendimento mais rápido do que está sendo passado pelo programa, isso agiliza a administração do banco de dados e ajuda a entender como as tabelas do banco de dados se comportam.

Foi criada uma nova abstração para representar a relação da quantidade de registros de uma dada tabela e a quantidade de registros de todas as tabelas visíveis pelo usuário do banco de dados: a metáfora de uma bateria.

Para construir a abstração do número de registros de uma tabela por meio de uma bateria foi criado primeiramente um *frame* para conter cada objeto envolvido no processo, esse *frame* contém o desenho de uma bateria (pilha) na cor vermelha, dentro desse desenho foi incluído um retângulo branco representando a quantidade faltante de carga na bateria (para que a carga total da bateria seja representada a altura do retângulo branco deve ser igual a zero) e para alterar a sua altura foi criado um método chamado *Resize*.

Esse *frame* foi incluído no formulário do programa, assim é possível controlar a "carga da bateria" mudando o tamanho da altura do retângulo branco. Na Figura 3.12 é possível visualizar o objeto que representa o retângulo branco, ele aparece selecionado.

O código implementado para alterar o tamanho desse objeto esta na Figura 3.13.

```
var Battery: TBattery;

{ TBatery }

{ TBatery }

procedure TBattery.Resize(Height: Integer);
begin
  Level.Height := Height;
end;

end.
```

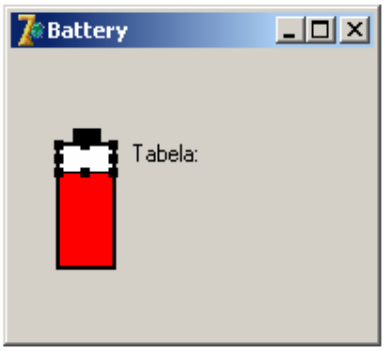

Figura 3.12 – Frame com os objetos que compõem a bateria.

```
procedure TfrmPLEasy.miTamanhoClick(Sender: TObject);
var aux: Integer;
begin
  if (ObjectTree.Selected.ImageIndex = IMG_TABLE) and
     (ObjectTree.Selected.Text <> 'Tabelas') then
  begin
     USER_TABLE.First;
     while not USER_TABLE.Eof do
     begin
        QueryRecCount.Close;
        QueryRecCount.SQL.Text := 'SELECT COUNT(*) "RecNo" FROM ' + USER_TABLETABLE_NAME.AsString;
        QueryRecCount.Active := True;
        aux := aux + QueryRecCount.FieldByName('RecNo').AsInteger;
        USER_TABLE.Next;
     end;

     with Bat do
     begin
        tbEditarDados.Close;
        tbEditarDados.TableName := ObjectTree.Selected.Text;
        tbEditarDados.Open;
        Resize(63 - (tbEditarDados.RecordCount * 63) div aux);
     end;
  end;
end;
```

Figura 3.13 – Trecho do programa que altera a carga da bateria.

Para saber em quantas partes dividir a altura da carga para cada registro da tabela relacionada em relação a todos os registros das tabelas do usuário é feito o cálculo representado na Figura 3.13.

```
procedure TfrmPLEasy.miTamanhoClick(Sender: TObject);
var i, j: Integer;
begin
  j := 0;
  for i := 1 to USER_TABLE.RecordCount do
  begin
    j := j + USER_TAB_COLS.RecordCount;
  end;
  if (ObjectTree.Selected.ImageIndex = IMG_TABLE) and (ObjectTree.Selected.Text <> 'Tabelas') then
  begin
    gdTableGridDblClick(Sender);
    with Bat do
    begin
      tbEditarDados.Close;
      tbEditarDados.Open;
      Resize(63 - (tbEditarDados.RecordCount * 63) div j);
    end;
  end;
end;
```

Figura 3.13 - Trecho de código para calcular a visualização

Analisando a Figura 3.14 pode-se ver que a variação se dá pela modificação da altura do objeto que representa o retângulo.

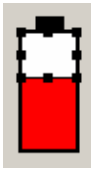

Figura 3.14 - Variação da carga da bateria

3.5 Visualização de Informação – tamanho do BD

Uma outra opção de visualização de informação disponibilizada pela ferramenta é um comparativo geral das quantidades de registros das tabelas por meio de um gráfico de barras.

Nesse gráfico, cada barra representa o número de registros de cada tabela do banco de dados. Na Figura 3.15 é mostrado o gráfico usando um banco de dados de exemplo e na Figura 3.16 é mostrado o trecho de código que gera as barras.

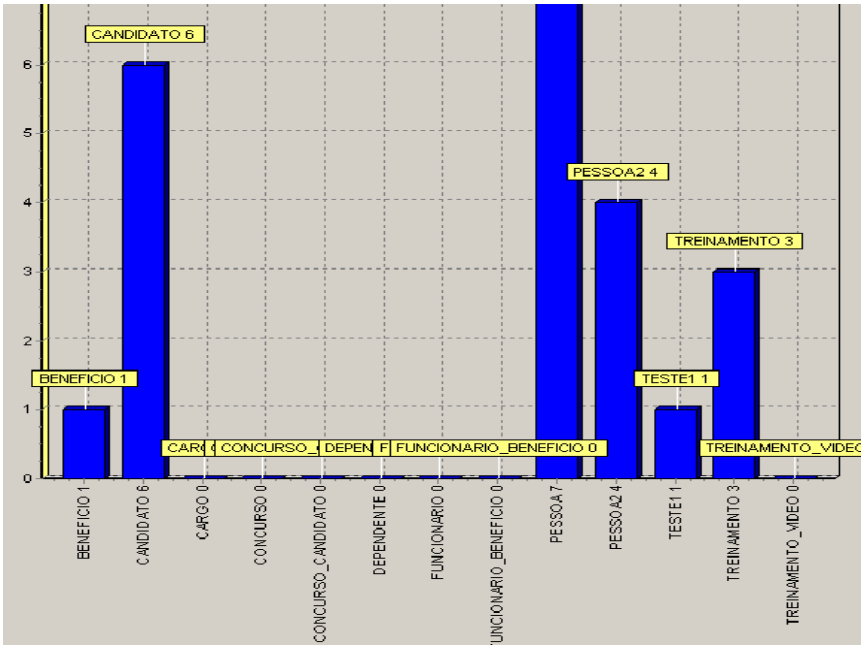

Figura 3.15 - Gráfico em barras do número de registros das tabelas do banco de dados.

```
USER_TABLE.First;
while not USER_TABLE.Eof do
begin
   QueryRecCount.Close;
   QueryRecCount.SQL.Text := 'SELECT COUNT(*) "RecNo" FROM ' + USER_TABLETABLE_NAME.AsString;
   QueryRecCount.Active := True;

   Series1.Add(QueryRecCount.FieldByName('RecNo').Value, USER_TABLETABLE_NAME.AsString + ' ' +
                                        QueryRecCount.FieldByName('RecNo').AsString, clBlue);
   USER_TABLE.Next;
end;
```

Figura 3.16 - Trecho de código que gera as barras

4 Estudo de casos

A ferramenta PLEasy, desenvolvida neste trabalho está apta á reconhecer objetos do banco através de uma árvore de dados, editar e executar scripts DDL, DML e de consulta (*selects*). É capaz também de editar os dados de uma tabela selecionada e mostrar através da metafora de uma bateria a quantidade de registros dessa tabela.

Foram realizados dois estudos de casos com a ferramenta usando-se dois bancos de dados diferentes e avaliando: árvore de dados, o editor/executor de *scrips*, a edição dos dados de uma tabela e, por fim, a metáfora da bateria.

O capítulo foi dividido em 3 tópicos: Árvore de Objetos, Editor/Executor e Metáfora da bateria.

4.1 Árvore de Objetos

Para testar a árvore de objetos foram observadas tabelas de bases diferentes, nas Figuras 4.1 e 4.2 são mostradas as duas árvores de objetos:

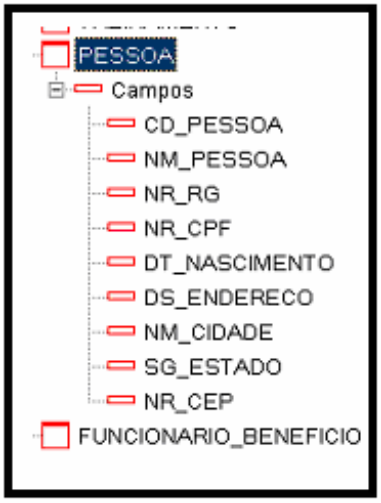

Figura 4.1 – Árvore de objetos para PESSOA do usuário banco de dados *ORCL*

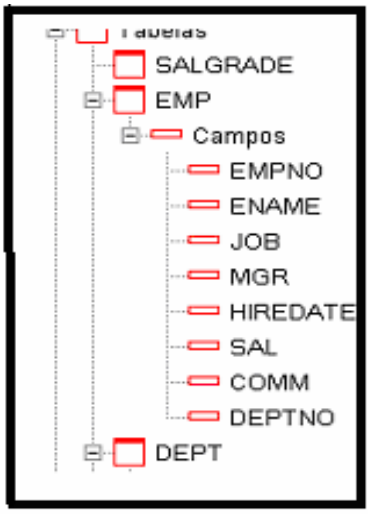

Figura 4.2 – Arvore de objetos para a tabela EMP do banco de dados *Scott*

No banco *ORCL* apareceram todos os campos como era esperado na tabela PESSOA representada pela Figura 4.1, isso também pode ser visualizado na tabela EMP do banco de dados *scott*.

A árvore de objeto reproduz os elementos do SGBD como uma hierarquia de elementos, é costume usar a hierarquia para organizar empresas, família, prioridades, e em muitas outras situações.

4.2 Editor/Executor

O primeiro teste com o editor/executor deste programa foi feito na base de exemplo do banco de dados no usuário *scott*. Para realizar o teste foi criada uma tabela baseada na estrutura de dados da tabela *BONUS* de acordo com a Figura 4.3.

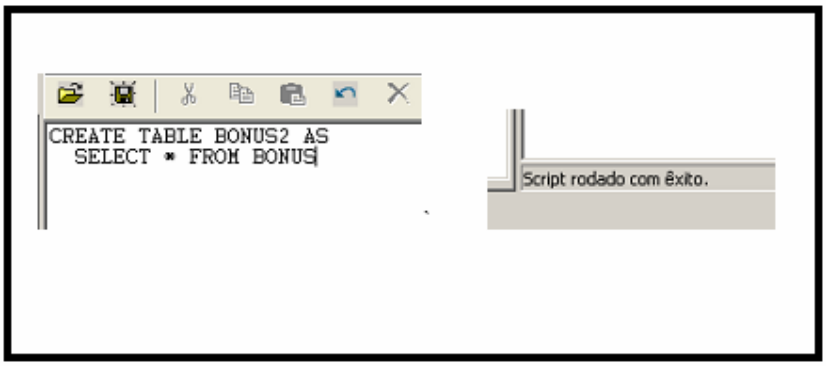

Figura 4.3 - *Script* executado no banco de dados exemplo.

Para verificar se a ferramenta criou a tabela BONUS2 foi usado o programa *SQL *Plus Worksheet* disponível pelo SGBD *Oracle*. Com a instrução "**SELECT * FROM TABS"** é possível exibir as tabelas do banco de dados como mostrado na Figura 4.4 e as tabelas aparecem na Figura 4.5..

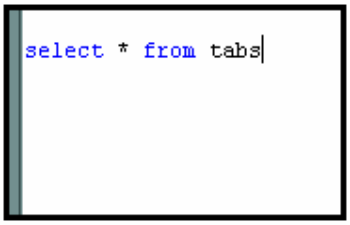

Figura 4.4 – Pesquisando tabelas do usuário scoot

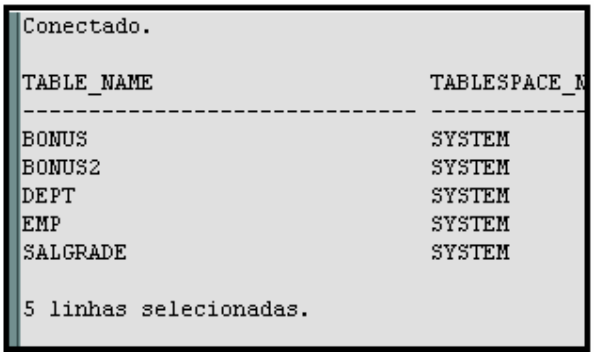

Figura 4.5 – Resultado da pesquisa na base scoot

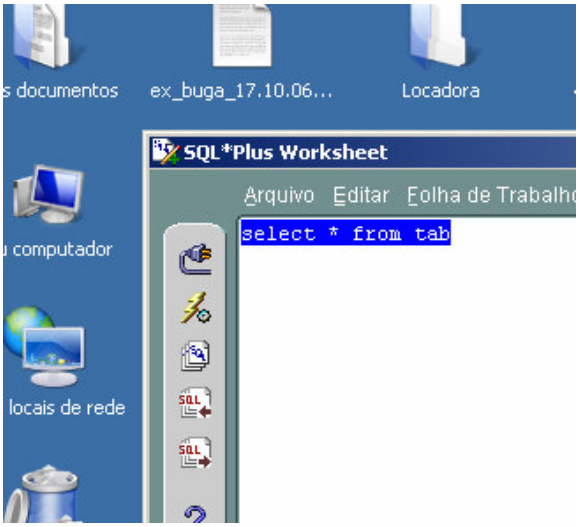

Figura 4.6 – Seleção de todas as tabelas do banco *SQL *Plus Worksheet*

O segundo teste foi feito no banco de dados *ORCL* e foi criada uma tabela baseada na estrutura de dados da tabela PESSOA* e o resultado. Essa operação é mostrada na Figura 4.6.

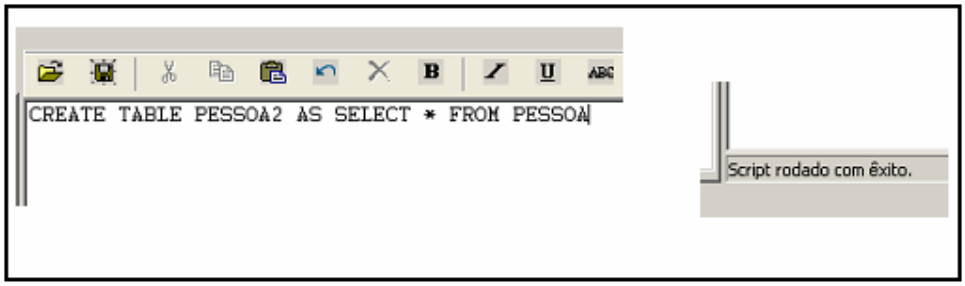

Figura 4.7 - *Script* executado no banco de dados exemplo.

Novamente, para verificar se a ferramenta criou a tabela PESSOA2 foi usado o programa *SQL *Plus Worksheet* disponível pelo SGBD *Oracle* como pode ser visto na Figura 4.8.

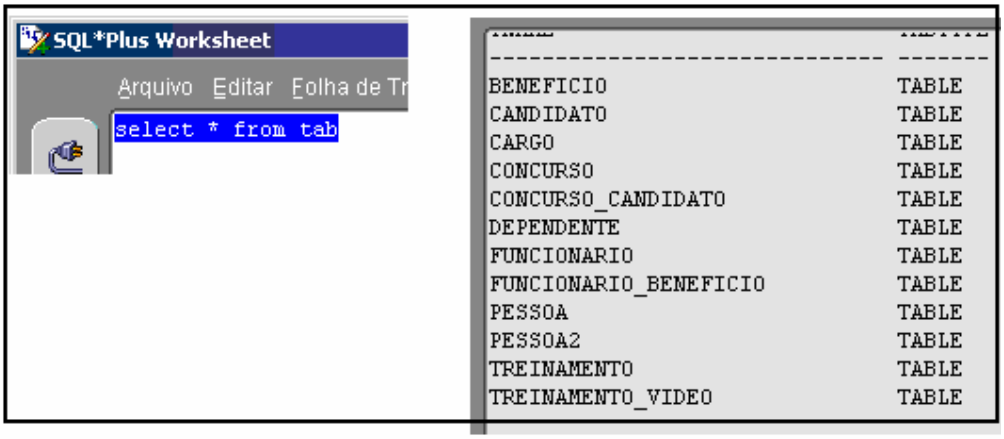

Figura 4.8 - Seleção das tabelas do banco de dados *ORCL*.

4.3 Metáfora da bateria

Para testar a metáfora da bateria no banco de dados *scott* foi usada a tabela DEPT como é observado na Figura 4.9 Nessa Figura pode ser observada a representação da quantidade de registros na tabela em relação ao desenho da carga da bateria.

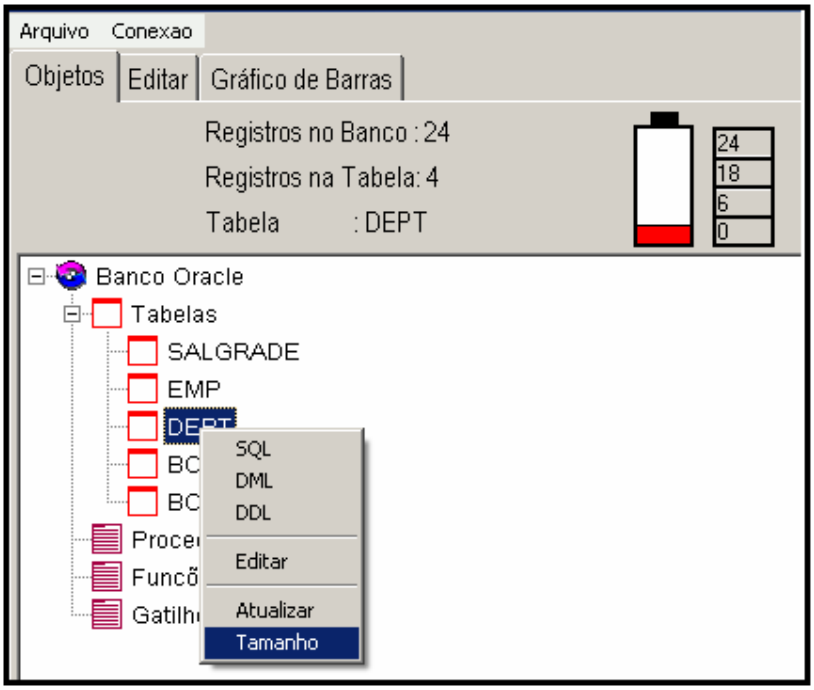

Figura 4.9 - Teste da metáfora da tabela DEPT do banco de dados de exemplo do *Oracle*

O teste da metáfora também foi feito com a tabela EMP do usuário *scoot* do banco de dados de exemplo do *Oracle*, a Figura 4.10.

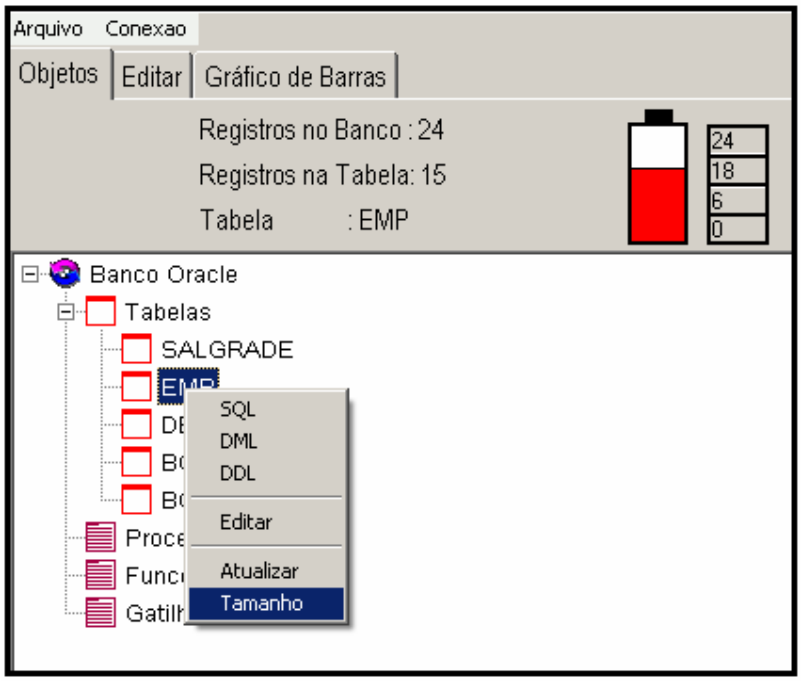

Figura 4.10 – Teste da metáfora da tabela EMP do banco de dados scott

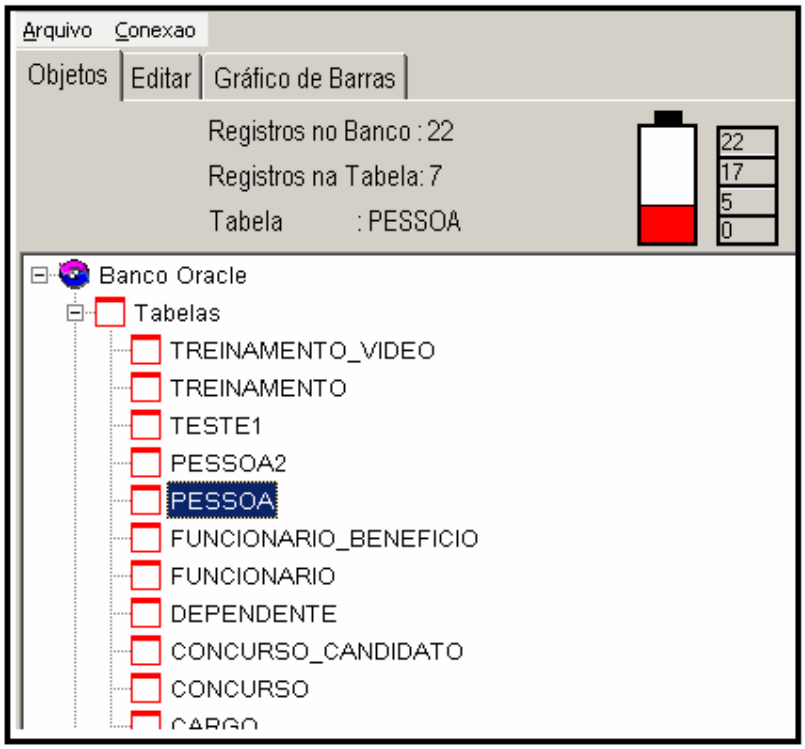

Figura 4.11 – Teste da metáfora da tabela PESSOA do banco de dados *ORCL*

Para testar a metáfora da bateria ainda foi feito um teste em uma tabela do usuário *orcl* do banco de dados *ORCL.* Na Figura 4.11 é mostrado o teste.

Conclusões

O crescimento do uso de SGBD(s) tornou necessário o uso de técnicas de visualização para proporcionar uma compreensão mais rápida e eficaz. A Ferramenta *PLEasy* possui algumas dessas técnicas implementadas.

Como mostra o estudo de caso apresentado neste trabalho, a árvore de objetos tornou-se muito útil para visualizar a estrutura do banco de dados. Ela foi usada para reconhecer os dados usados durante esses testes.

A idéia de desenhar a imagem de uma bateria e abstrair a quantidade de registros de uma dada tabela por meio da visualização de sua carga, foi atingido um diferencial de outras ferramentas disponíveis no mercado.

O editor/executor atingiu as expectativas na execução de comandos DDL, DML e em pesquisas SQL.

Os testes de criação, manipulação e seleção de objetos obtiveram os resultados esperados. Isso faz da ferramenta uma opção viável para o auxílio a programadores e administradores de bancos de dados.

Algumas sugestões para trabalhos futuros são completar a árvore de objetos permitindo a visualização de outros objetos do SGBD como, por exemplo, chaves estrangeiras, *constraints* e packages. Outra idéia é aprimorar o módulo Editor/Executor da ferramenta incluindo a criação de *templates* e o reconhecimento da sintaxe *SQL* do *Oracle* nos *scripts* e avisar sobre erros antes de sua execução. Também é possível gerar outras metáforas para visualização de dados para o SGBD.

Referências

ASADI, M. **Oracle 7.3** Indianapolis: Sams Publishing, 1997.

CARD, S. K.; MACKINLAY, J. D.; SHNEIDERMAN, B. **Information Visualization: Using Vision to Think**. San Diego: Academic Press, 1999.

CANTÚ, M. **Dominando o Delphi 5 – A Bíblia.** São Paulo: Makron Books, 2000

CHEN, C.. **Information Visualization and Virtual Environments**. Singapore: Springer, 1999.

FREITAS, C. M. D. S, CHUBACHI, O. M, Luzzardi, P.R.G.; CAVAL R.A.; **Introdução à Visualização de Informações.** In: Revista Rita, v.8, n.2, p1-16, 2001. Disponível em: <http://www.inf.ufrgs.br/~revista/docs/rita08/rita_v8_n2_p143a158.pdf>. Acesso em 18 nov. 2006.

HARRINGTON, J. L. **Projetos de bancos de dados relacionais : teoria e prática.** Sl.**,** 2002.

KORTH, Henry F.; SILBERSCHATZ, Abraham; SUDARSHAN, S.. **Sistema de banco de dados**. 3ª ed. São Paulo: Makron Books, 1999. 778p.

MACHADO, F. N. R. **Bancos de Dados: Projeto e Implementação.** São Paulo: Ed. Erica 2004

NETO, M. C. M., Mendonça, M., SANTOS C. A. S.; **GraphMiner: Uma Ferramenta de Mineração Visual de Dados em Bases Relacionais.** 2006. In: Webmedia & LA Web, 2004. Disponível em: <http://is.ls.fi.upm.es/jiisic04/Papers/72.pdf>. Acesso em 16 nov. 2006.

ORACLE**; Sobre o Oracle.** Disponível em:

<http://www.oracle.com/global/br/corporate/index.html>. Acesso em: 28 nov. 2006.

TRAINA, A. J. M., TRAINA, C. J.; BOTELHO, E.; BARIONE, M. C. N.; BUENO, R. **Visualização de dados em Sistemas de Bases de Dados Relacionais.** In: XVI Simpósio Brasileiro de Banco de Dados. Anais. Rio de Janeiro/RJ: COPPE/UFRJ, outubro de 2001. p. 95-109.

WIKKIPEDIA; **Oracle.** Disponível em:

<http://pt.wikipedia.org/wiki/Oracle>. Acesso em 28 nov. 2006.