UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

FERRAMENTA DE GERENCIAMENTO PARA O BANCO DE DADOS FIREBIRD

CARLOS EDUARDO WERNER

BLUMENAU
2004

2004/1-03

CARLOS EDUARDO WERNER

FERRAMENTA DE GERENCIAMENTO PARA O BANCO DE DADOS FIREBIRD

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Alexander Roberto Valdameri

BLUMENAU
2004

2004/1-03

FERRAMENTA DE GERENCIAMENTO PARA O BANCO DE DADOS FIREBIRD

Por

CARLOS EDUARDO WERNER

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Alexander Roberto Valdameri
Membro:	Prof. Evaristo Baptista, FURB
Membro:	Prof. Marcel Hugo, FURB

Blumenau, 01 de junho de 2004

Dedico este trabalho a todos os amigos, especialmente aqueles que me ajudaram diretamente na realização deste. Em memória de minha mãe Cláudia Werner por sempre acreditar no meu potencial e ter sempre me incentivado para a realização do mesmo.

Não estamos aqui para sobreviver e sim para explorar a oportunidade de vencer adquirindo o saber.

Renato da Costa

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha família, por terem me dado muita força durante esta jornada.

Aos meus amigos que eu fiz ao longo destes anos e que me ajudaram em tudo o que precisei para concluir cada fase deste curso.

Ao meu orientador Prof Alexander Roberto Valdameri por toda a atenção e incentivo na orientação disponibilizada para o desenvolvimento deste trabalho.

Agradeço a todos que ajudaram diretamente e indiretamente na minha formação universitária, tornando a universidade mais culta e amistosa.

RESUMO

Atualmente os Sistemas Gerenciadores de Bancos de Dados (SGBD) *Open Source* apresentam uma carência em termos de ferramentas de administração. Considerando este fato, este trabalho apresenta o desenvolvimento de uma ferramenta para o sistema de gerenciamento do banco de dados Firebird, tendo como seu principal objetivo gerenciar bases de dados, permitindo manter tabelas bem como os seus campos. Para isto, foram estudadas as técnicas do ambiente de programação Delphi e o SGBD Firebird. Esta ferramenta tem o intuito de suprir a necessidade de ferramentas *Open Source*.

Palavras chaves: Firebird; SGBD; SQL; Ferramentas de Gerenciamento.

ABSTRACT

Nowadays, Open source database management systems (DBMS) show a lack in terms of administration tools. Introducing this, the present research demonstrates the development of a tool to the management system of a Firebird Data Bank, having as the main goal the ability to database manage, making possible to keep tables as well as their fields. In order to do so, the techniques of the programming Delphi and DBMS Firebird were studied. This tool has the purpose of supplying the need of the Open Source tools.

Key-Words: Firebird; DBMS; SQL; Management tools.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

DBA – DataBase Administrator

DDL – Data Definition Language

SGBD – Sistema de gerenciamento de bancos de dados

LISTA DE SÍMBOLOS

$ - cifrão

SUMÁRIO

1 INTRODUÇÃO

Este capítulo fornece uma visão geral do assunto abordado neste trabalho. Na contextualização ainda, tem-se o relato dos principais objetivos e a estrutura do trabalho.

1.1 CONTEXTUALIZAÇÃO

O interesse das empresas desenvolvedoras de softwares por sistema de gerenciamento de bancos de dados que sejam Open Source (fonte aberto) tem crescido nos últimos anos. Atualmente os mais conhecidos são: SAP DB, MySQL, PostGreSQL e o Firebird.

Conforme Date (2000, p. 07), os SGBDs são uma coleção de ferramentas que possibilitam criar e manter um banco de dados, em outras palavras, é a ferramenta que trata todo e qualquer acesso ao banco de dados.

O Firebird é considerado um dos SGBDs que mais tem evoluído em termos de funcionalidade e utilização mercadológica (Freitas, 2002). Dentre as principais razões que caracterizam o Firebird com uma boa solução pode-se declarar o custo (código aberto).

O Firebird também dá continuidade à longa tradição do SGBD Interbase. Os direitos autorais do SGBD Interbase são pertencentes à empresa Borland. O Firebird possui cinco plataformas que são bem suportadas pelo mesmo: Win32, Linux, Solaris (Sparc e X86) e o Mac OS X. Além dessas, existem mais cinco plataformas que são suportadas menos ativamente: FreeBSD, NetBSD, AIX, HP-UX e SCO.

Todavia, o SGBD Firebird não apresenta em sua estrutura de funcionamento uma ferramenta com o seu código fonte aberto de fácil manuseio para futuras atualizações da mesma. Neste sentido, este trabalho apresenta o desenvolvimento de uma ferramenta, cuja funcionalidade é facilitar a interação entre o usuário (administrador de banco de dados) e o banco de dados, oferecendo em um ambiente gráfico as funcionalidades para manutenção de estruturas e dados.

Vislumbra-se a possibilidade da utilização da ferramenta proposta em ambiente acadêmico nas disciplinas de banco de dados, tendo como objetivo suprir a carência de interfaces interativas e de fácil acesso.

1.2 OBJETIVOS

O objetivo deste Trabalho de Conclusão de Curso é desenvolver uma ferramenta que permite gerenciar o SGBD Firebird.

Os objetivos específicos do trabalho são:

a) permitir a manutenção da estrutura do metadados;
b) permitir a inclusão/alteração/exclusão de dados na estrutura da base de dados;
c) permitir ao usuário executar comandos SQL;
d) permitir a manipulação de uma base de dados através de uma interface interativa, utilizando os conceitos de navegação do Windows (campos de edição, *grids*, botões, entre outros).

1.3 ESTRUTURA DO TRABALHO

Este trabalho está dividido em forma de capítulos descritos a seguir.

O primeiro capítulo expõe na introdução uma justificativa do que originou este trabalho como também uma síntese do que será tratado no desenvolvimento do trabalho. Os objetivos a serem alcançados.

O segundo capítulo apresenta o SGBD Firebird, suas principais características e funcionalidades.

O terceiro capítulo faz uma breve exposição das ferramentas IBExprt e Ibconsole que são algumas interfaces de gerenciamento para o SGBD Interbase.

O quarto capítulo apresenta a especificação feita para o desenvolvimento do trabalho e apresenta a ferramenta com suas funcionalidades.

O quinto capítulo expõe as considerações finais após o desenvolvimento do trabalho e algumas sugestões para sua continuação.

2 BANCO DE DADOS

2.1 CONCEITO

Conforme Date (2000, p. 07), um sistema de banco de dados é basicamente um sistema computadorizado de armazenamento de registros; isto é, um sistema computadorizado cujo propósito geral é armazenar informações e permitir ao usuário buscar e atualizar essas informações quando solicitado. As informações em questão podem ter qualquer significado para o indivíduo ou a organização a que o sistema deve servir – em outras palavras, tudo o que seja necessário para auxiliar no processo geral da tomada de decisões de negócios desse indivíduo ou dessa organização.

Conforme WEB Information (2002), um banco de dados é uma coleção logicamente coerente de dados com determinado significado. Um banco de dados representará sempre aspectos do mundo real. Assim sendo, uma base de dados (ou banco de dados, ou ainda BD) é uma fonte de onde pode-se extrair uma vasta gama de informações derivadas, que possui um nível de interação com eventos como o mundo real que representa. A forma mais comum de interação usuário - banco de dados dar-se-á através de sistemas específicos que, por sua vez, acessam o volume de informações geralmente através da *Strutured Query Language* (SQL).

Conforme Mecenas (2000, p. 05), a linguagem SQL foi criada pela IBM como linguagem de acesso ao banco de dados relacional para *maiframes* . Ela atende ao conjunto completo de exigências para se classificar como linguagem relacional e foi padronizada para uso em qualquer plataforma de computadores: *mainframes*, minis e micros, de qualquer marca; é uma linguagem aberta para uso por qualquer fornecedor de software.

2.2 SGBD

Conforme Date (2000, p. 07), todas as solicitações de acesso ao banco de dados são tratadas pelo SGBD. Uma função geral fornecida pelo SGBD é, portanto, a de isolar os usuários do banco de dados dos detalhes do nível de hardware.

Uma característica importante do SGBD é manter não somente os dados, mas também a forma como os mesmos são armazenados, contendo uma descrição completa do banco de dados. Estas informações são armazenadas no catálogo do SGBD sendo denominada Metadados, o qual contém informações como a estrutura de cada arquivo, o tipo e o formato

de armazenamento de cada tipo de dado, restrições, etc. No processamento tradicional de arquivos, o programa que irá manipular os dados deve conter este tipo de informação, ficando limitado a manipular as informações que o mesmo conhece. Utilizando a abordagem banco de dados, a aplicação pode manipular diversas bases de dados diferentes.

O SGBD deve fornecer ao usuário uma "representação conceitual" dos dados, sem fornecer muitos detalhes de como as informações são armazenadas. Um "modelo de dados" é uma abstração de dados que é utilizada para fornecer esta representação conceitual utilizando conceitos lógicos como objetos, suas propriedades e seus relacionamentos. A estrutura detalhada e a organização de cada arquivo são descritas no catálogo.

Conforme Date (2000, p. 37), um SGBD deve possuir as seguintes funções: definição de dados, manipulação, otimização, segurança, integridade, recuperação, concorrência de dados e dicionário de dados.

O Firebird é considerado um SGBD relacional.

2.3 FIREBIRD

Este capítulo descreve um pouco da história do Firebird, trazendo também suas principais características, funcionalidades e também algumas informações sobre a sua instalação.

2.3.1 HISTÓRICO

O Firebird nasceu do InterBase 6.0 *Open Source*. Após a Borland abrir o código do InterBase na versão 6.0, ela decidiu que continuaria mantendo uma versão comercial do produto (com o código fechado). Atualmente um grupo de pessoas, algumas delas que já trabalhavam com o InterBase dentro da própria Borland, decidiram dar continuidade à versão aberta criando o Firebird. Hoje estão disponíveis o InterBase 6.0 (*Open Source*) que não sofreu mais atualizações por parte da Borland desde o release 6.0.2, o InterBase 7.1 que é um produto comercial desenvolvido pela Borland e as versões Firebird que são *Open Source* que estão crescendo e adquirindo novos recursos a cada dia. Atualmente, o Firebird é mantido por uma comunidade de desenvolvedores (denominado PHOENIX) e pela Fundação Firebird (Freitas 2002).

2.3.2 PRINCIPAIS CARACTERÍSTICAS

Ao contrário da Borland, que se mantém em silêncio no que diz respeito a novas implementações no InterBase, bem como se novos recursos serão disponibilizados na versão *Open Source* do banco ou se somente estarão disponíveis na versão certificada, o projeto Firebird é totalmente aberto à comunidade. Qualquer um pode se informar da situação atual do código, saber o que foi alterado e o que está por vir. A comunidade de desenvolvedores está aberta para a participação de todos que quiserem auxiliar no desenvolvimento do código, correção de *bugs*, documentação, desenvolvimento de ferramentas, etc.

Não existe, no entanto, uma comunidade aberta trabalhando no código atual do InterBase da Borland, portanto mesmo que alguém queira trabalhar nele, não encontrará muita gente com quem conversar, trocar idéias e experiências. A Borland mantém uma equipe de desenvolvimento interna trabalhando na versão certificada do banco mas, como dito anteriormente, não se sabe se as novas implementações estarão disponíveis numa próxima versão do InterBase.

No Firebird não existem cronogramas para *upgrades* e as correções de *bugs* são disponibilizadas quase que instantaneamente.

Conforme Rodrigues (2002), algumas características são apresentadas em nível teórico, ou seja, que é possível alcançar em termos do projeto do banco, mas não significa dizer que tais características estejam de fato implementadascomo :

a) Número máximo de clientes conectados a um servidor: embora esse número possa sofrer variações que depende de fatores de utilização do FIREBIRD, diz-se que 150 usuários possam ter acesso simultâneo ao banco de dados.
b) Tamanho máximo de um banco de dados: em se tratando de arquivo único (ou simples), um banco de dados pode chegar a 4 GB em ambiente Windows e na maioria dos ambientes UNIX, podendo atingir vários terabytes quando se utiliza arquivos múltiplos, ou seja, um arquivo principal e vários arquivos secundários (lembre-se que um banco pode ser composto de vários arquivos, sendo que o maior tamanho de um banco Interbase ou um banco Firebird reportado até hoje é de 980Gb). A Tabela 1 fornece uma indicação mais precisa dos valores para arquivos simples no ambiente Windows:

Tabela 1 – Tamanho de banco de dados para ambiente Windows

Sistema Operacional	Tamanho do Arquivo
Win9x/ME (FAT16)	2 GB
WinNT/2000 (FAT16)	4 GB
Win9x/ME (FAT16)	4 GB
WinNT/2000 (NTFS)	16.384 GB

c) Número máximo de tabelas por banco de dados: em termos de projeto, isto é, teoricamente, pode-se chegar a 65536 tabelas.
d) Tamanho máximo de uma linha: 64 Kb.
e) Número máximo de linhas e colunas por tabela: por projeto, esse número pode chegar a 2^{32} – 4.294.967.296.
f) Número máximo de índices por tabela: por projeto, esse número pode chegar a 65536. Efetivamente, o FIREBIRD permite até 64 índices por tabela. O número teórico, ainda que fosse possível, deixaria o banco com performance sofrível.
g) Número máximo de índices por banco de dados: Por projeto, esse número pode chegar a 2^{32}.

2.3.3 FUNCIONALIDADES

O Firebird oferece uma solução como sistema gerenciador de bancos de dados de alta performance além de hoje ser totalmente independente do Interbase. O mesmo enquadra-se nos requerimentos do padrão SQL-92, suportando integridade referencial declarativa com operações em cascata, atualizações de visões e junções externas. Já os servidores Firebird disponibilizam bibliotecas que suportam o desenvolvimento de aplicações com instruções SQL embutidas e aplicações com instruções SQL dinâmicas (DSQL).

Em qualquer plataforma, aplicações podem ser escritas utilizando chamadas diretas à camada API do Firebird, isto é, uma biblioteca de funções que permite enviar requisições que realizam operações nos bancos de dados do servidor.

O servidor Firebird está estruturado em um modelo de transação. As transações permitem que instruções enviadas ao banco de dados ocorram em blocos com a característica especial de tudo ou nada. Significa dizer que se um conjunto de instruções pode ser tratado com uma única instrução, com possibilidade de ser executada completamente (em caso de sucesso) ou de não ser executada (insucesso).

O gerenciamento de transações do Firebird habilita aplicações clientes a iniciarem múltiplas transações simultâneas. Há um complexo e explícito controle sobre as transações apresentadas ao servidor. As transações podem ser isoladas de modo a não serem afetadas por alterações realizadas por transação concorrentes.

Além disso, segundo Rodrigues (2002), o Firebird oferece excelente concorrência de acesso, alta performance, uma poderosa linguagem com suporte a *stored procedures* (é um objeto de banco de dados que contém comandos que se resumem em linhas de comando que processam tarefas, ou fazem chamadas a outras *procedures*) e *triggers* de bancos (são blocos de sentenças SQL que contêm códigos que implementam instruções que podem afetar dados contidos em tabelas).

Conforme Rodrigues (2002), este SGBD dispensa maiores estruturas dentro de uma empresa, bastando instalar o software e usá-lo, sem a interferência freqüente de profissionais especializados na manutenção do banco de dados de produção.

Acompanhando isso tudo, o mesmo ainda dispensa o uso de super-servidores (máquinas robustas), usando pouco espaço em disco para sua instalação e utilizando pouca memória em situações normais de uso. Por isso, a plataforma necessária para a sua instalação e utilização pode ser reduzida, diminuindo consideravelmente os custos do projeto.

2.3.4 INSTALAÇÃO

Conforme Cantu (2002), o Firebird é conhecido pela sua facilidade de instalação e configuração. Esses sempre foram pontos altos nunca igualados por qualquer outro SGBD. Além disso, o Firebird é um dos menores servidores de bancos de dados que existem em termos de kBytes, ocupando pouco espaço em disco.

Segundo Mecenas (2000, p. 08), dois aplicativos essenciais são utilizados para o funcionamento pleno do Firebird, o Firebird Server Control e o Firebird Guardian.

O Firebird Server Control é o servidor do Firebird. No sistema operacional Windows NT, o Firebird Server Control é executado como um serviço e nos sistemas Windows 95/98 e Me, como uma aplicação.

O Firebird Guardian tem como finalidade tentar manter ativo o Firebird Server Control quando este é submetido a interrupções anormais em seu funcionamento.

2.3.5 METADADOS DO FIREBIRD

Os SGBD possuem uma série de informações que servem para manter a sua estrutura através das relações entre as diversas tabelas, denominado metadados ou dicionário de dados. Nestas tabelas estão armazenados diversos tipos de informações, entre elas pode-se citar: os nomes das relações, os nomes dos atributos de cada relação, os domínios de atributos, nomes de visões definidas no BD com suas definições, as restrições de integridade para cada relação, podendo ainda armazenar nomes dos usuários autorizados a realizar consultas ou atualizações, número de tuplas máximo para cada entidade.

As tabelas de sistema começam sempre com o prefixo RDB$ e somente os objetos que fazem parte do Firebird podem ter seus nomes começando com esse prefixo.

Nas Figuras 1 e 2 pode-se observar algumas das tabelas (as consideradas relevantes para a execução deste trabalho) que compõem o metadados do Firebird não havendo o relacionamento explícito entre as tabelas, uma vez que este é mantido de forma implícita pelo SGBD.

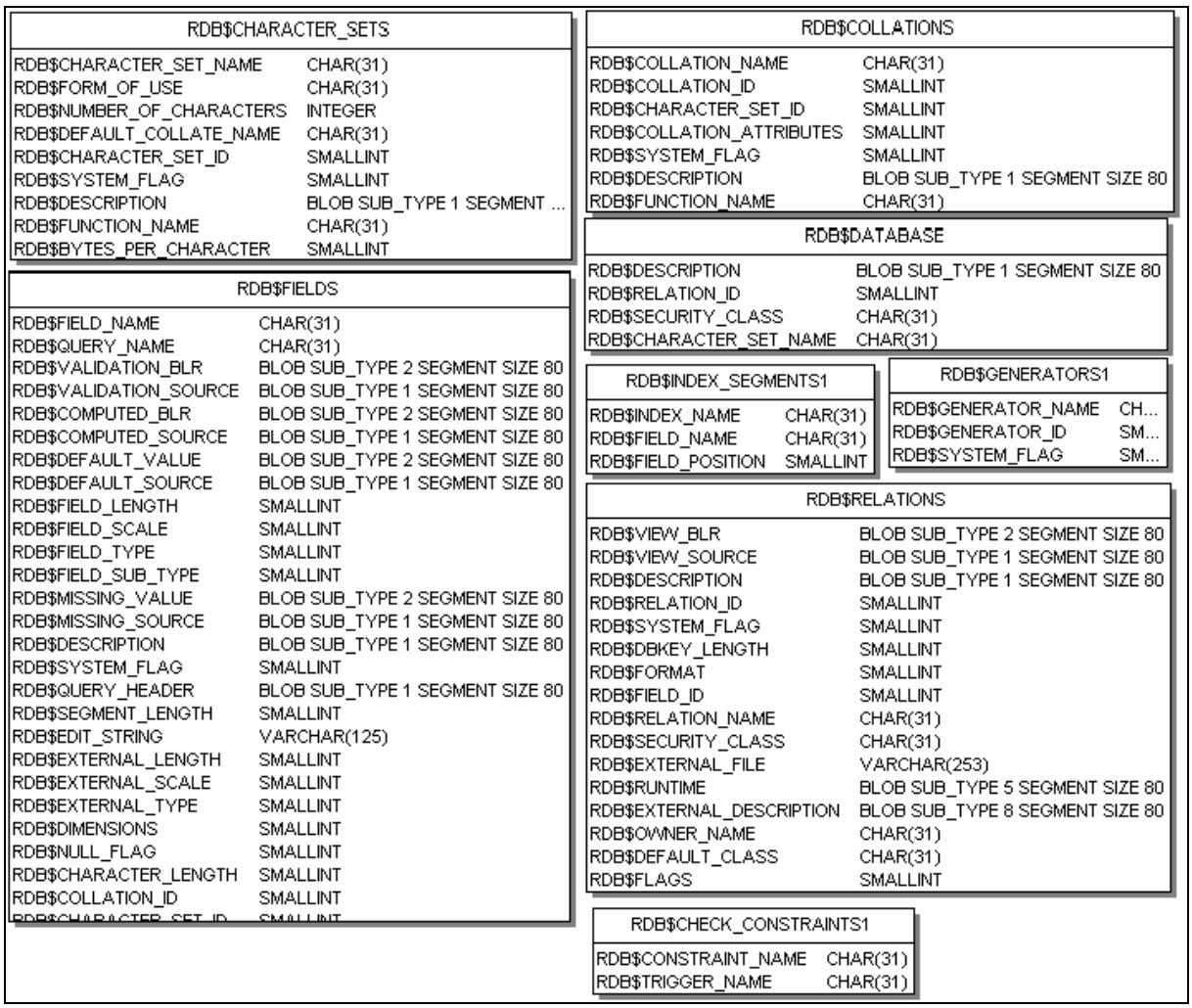

Figura 1 – Tabelas do metadados

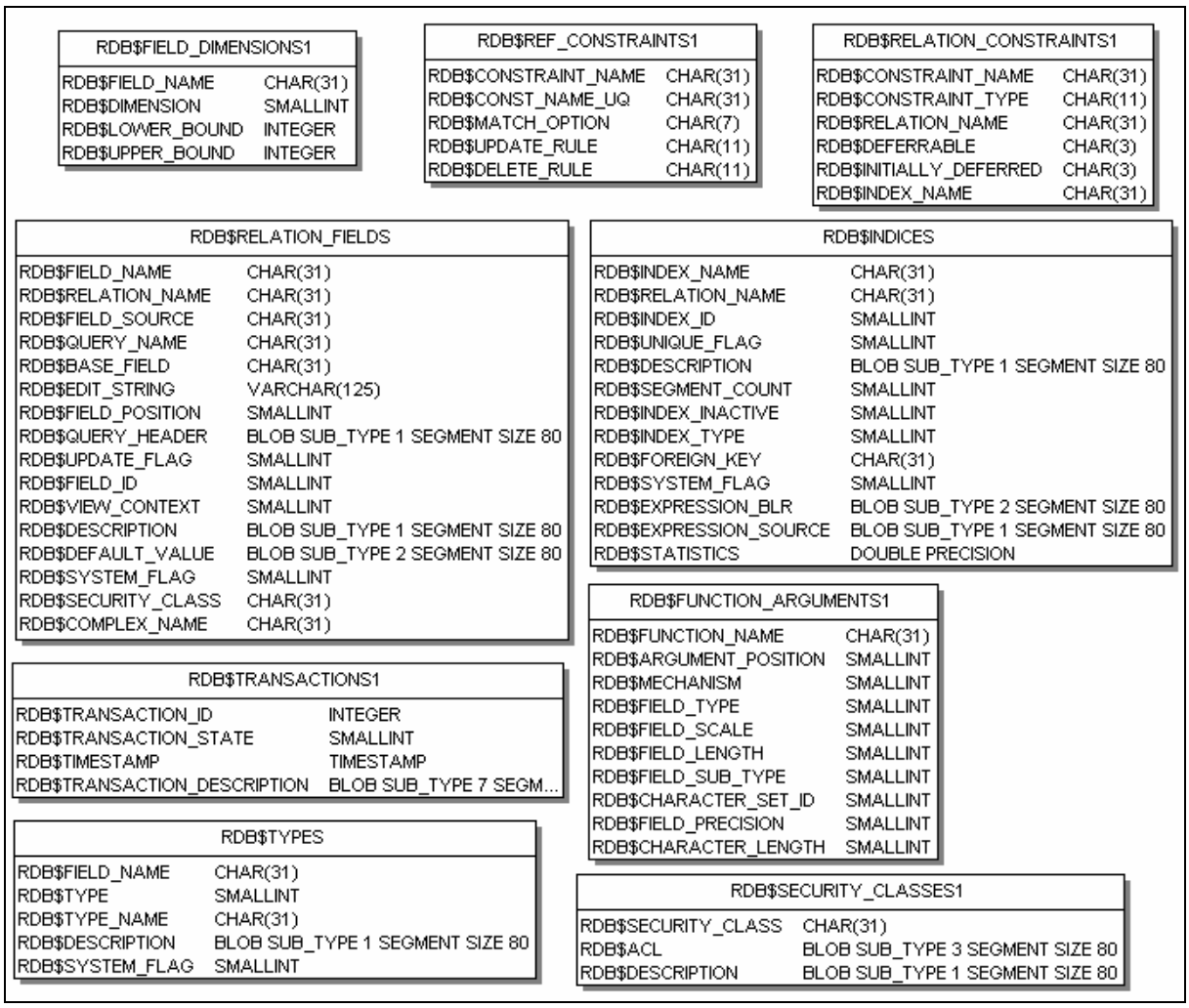

Figura 2 – Continuação das tabelas do metadados

A Tabela 2 descreve as estruturas mapeadas a partir do metadados do Firebird, onde é descrito o seu nome é a sua descrição:

Tabela 2 – Descrição do metadados do Firebird

Tabela	**Descrição**
RDB$CHARACTER_SETS	Descreve os grupos de caracteres válidos no BD.
RDB$COLLATIONS	Lista os COLLATES disponíveis ao usuário no BD.
RDB$CHECK_CONSTRAINTS	Armazena os dados de integridade referencial definidos pelas CHECKs e pelo uso da opção NOT NULL.
RDB$DATABASE	Define o Banco de Dados propriamente dito.
RDB$DEPENDENCIES	Armazena as dependências de tabelas e campos com outros objetos do BD, como por exemplo *VIEWS*, TABELAS, *TRIGGERS*, campos calculados. O Firebird utiliza essa tabela para checar se um objeto pode ser removido caso ele não tenha nenhuma dependência.
RDB$EXCEPTIONS	Define as exceções geradas por erros, incluindo as exceções definidas pelo usuário.
RDB$FIELD_DIMENSIONS	Contém as dimensões das colunas do tipo ARRAY
RDB$FIELDS	Define as características de um campo. Todo campo ou domínio tem um registro correspondente nessa tabela.
RDB$FILES	Define os arquivos de Shadow do BD bem como os arquivos secundários que podem compor o BD.
RDB$FILTERS	Contém informações sobre os filtros de BLOB.
RDB$FORMATS	Mantém informações sobre os formatos das colunas das tabelas. Cada vez que o tipo/formato de um campo é alterado, o Firebird atribui um novo número de formato à tabela, permitindo que os aplicativos continuem acessando essa tabela sem que sejam recompilados.
RDB$FUNCTION_ARGUMENTS	Define os atributos dos argumentos de uma função.
RDB$FUNCTIONS	Armazena a definição das UDFs (Funções Definidas pelo Usuário)
RDB$GENERATORS	Armazena as informações sobre os GENERATORs.
RDB$INDEX_SEGMENTS	Especifica os campos que compõem um índice de uma tabela. Permite a alteração de registros nessa tabela sem que sejam apagados e recriados os registros na tabela RDB$INDICES dentro da mesma transação podendo gerar corrupção no BD.
RDB$INDICES	Define as estruturas dos índices que permitem que o Firebird localize registros de maneira eficiente. Cada índice definido nessa tabela deve ter um registro associado na tabela RDB$INDEX_SEGMENTS.

RDB$PAGES	Mantém uma relação de páginas alocadas para o BD. É uma tabela que deve ser manipulada com cuidado, pois se for alterada provoca corrupção no BD.
RDB$PROCEDURE_PARAMETERS	Armazena informações sobre cada parâmetro definido em cada *Stored Procedure* do BD.
RDB$PROCEDURES	Mantém as informações sobre as *Stored Procedures* do BD.
RDB$REF_CONSTRAINTS	Armazena as informações sobre a Integridade Referencial no BD.
RDB$RELATION_CONSTRAINTS	Armazena informações sobre Integridade Referencial das Tabelas do BD.
RDB$RELATION_FIELDS	Mantém a lista dos campos das tabelas e as informações das características de uma coluna para os domínios. Os nomes dos campos são relacionados através da coluna RDB$FIELD_SOURCE, com um registro na tabela RDB$FIELDS que contém um nome de sistema ("SQL$<n>"). Os dados armazenados incluem informações sobre os tipos das colunas, e podem conter informações sobre os valores default e possibilidade de NULLs.
RDB$RELATIONS	Define algumas das características das TABELAS e *VIEWS*. As outras características são armazenadas na tabela RDB$RELATION_FIELDS.
RDB$ROLES	Armazena os ROLES que foram definidos no BD, bem como o criador de cada ROLE.
RDB$SECURITY_CLASSES	Armazena as listas de controle de acesso associando-as com as tabelas, *views*, campos, BD. As informações sobre os objetos SQL são duplicadas na tabela RDB$USER_PRIVILEGES.
RDB$TRANSACTIONS	Mantém a lista das transações entre múltiplos Bancos de Dados.
RDB$TRIGGER_MESSAGES	Define as mensagens dos *triggers* relacionando-as com cada *trigger*.
RDB$TRIGGERS	Define os *TRIGGERs*.
RDB$TYPES	A capacidade dessa tabela não é utilizada na versão atual do Firebird. Ela armazena tipos de dados enumerados e apelidos para charsets e collates.
RDB$USER_PRIVILEGES	Mantém as informações sobre os privilégios atribuídos aos usuários através do comando GRANT.

O próximo capítulo apresenta duas ferramentas proprietárias para o gerenciamento do SGBD Firebird.

3 FERRAMENTAS PARA O GERENCIAMENTO DO FIREBIRD

Atualmente existe uma certa carência de ambientes para o gerenciamento de SGBDs, em especial os *Open Source*, dos quais pode-se citar principalmente o Firebird. Tal carência se dá em virtude dos grupos de desenvolvedores em SGBDs *Open Source* enfatizar o núcleo do SGBD e não as ferramentas administrativas para os mesmos.

Para ser considerada uma ferramenta de boa qualidade a mesma deverá possuir principalmente as seguintes características:

a) configurar e administrar o servidor Local ou Remoto;
b) digitar e executar comandos interativos SQL;
c) registrar e criar banco de dados;
d) fazer a manipulação de dados: SELECT, INSERT, UPDATE e DELETE;
e) administrar a segurança;
f) excluir todas as informações de um banco de dados.

Existem algumas ferramentas de interface gráfica para o SGBD Firebird no mercado, como por exemplo, o IBConsole e o IBExpert.

3.1 IBCONSOLE

O IBConsole é um utilitário gráfico idealizado pela Borland para ser uma ferramenta de administração do InterBase/Firebird tanto em nível de dados quanto em nível de suporte e operação, verificação de planos de acesso e ainda para configuração, criação e manutenção de banco de dados. O IBConsole é uma ferramenta que acompanha a instalação do Interbase e não possui o seu código fonte aberto, agregando assim um valor comercial.

Para demonstrar a funcionalidade do IBConsole, seguem algumas ilustrações.

Na Figura 3 é apresentado a tela principal do IBConsole.

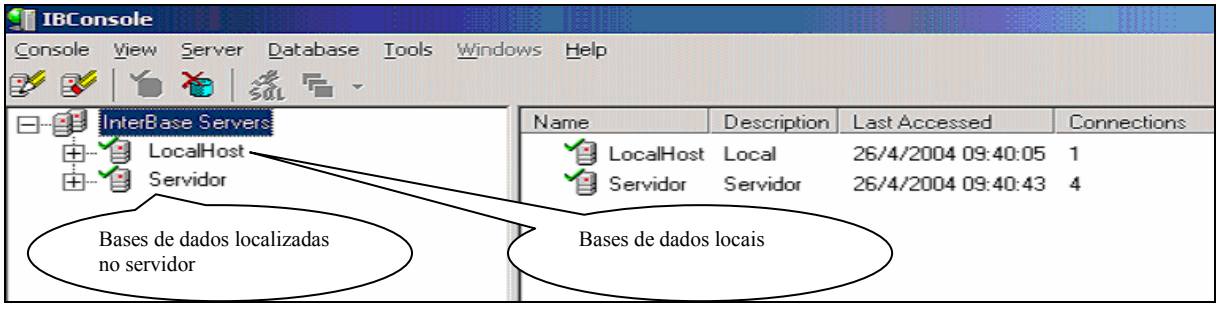

Figura 3 – Tela principal do IBConsole

Na Figura 4 pode-se observar a tela de criação de uma base de dados local utilizando o nome de “teste.fdb”.

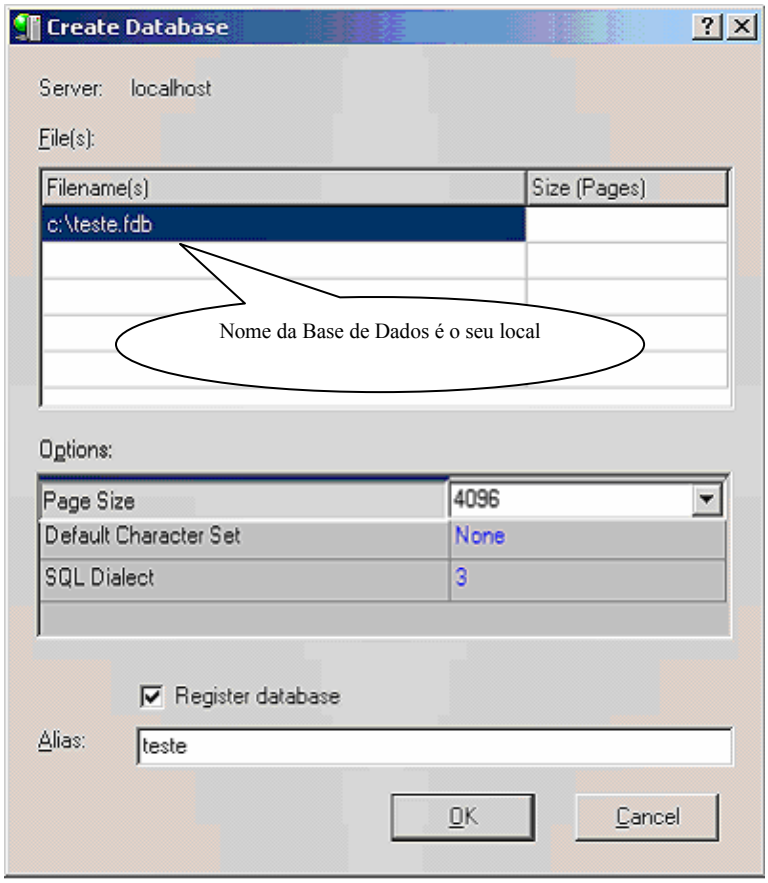

Figura 4 – Criação da base de dados pelo IBConsole

A Figura 5 ilustra a criação de uma tabela no IBConsole, apresentando especificamente a criação dos campos. No IBConsole a criação de tabela só é feita através da submissão da execução de *scripts*.

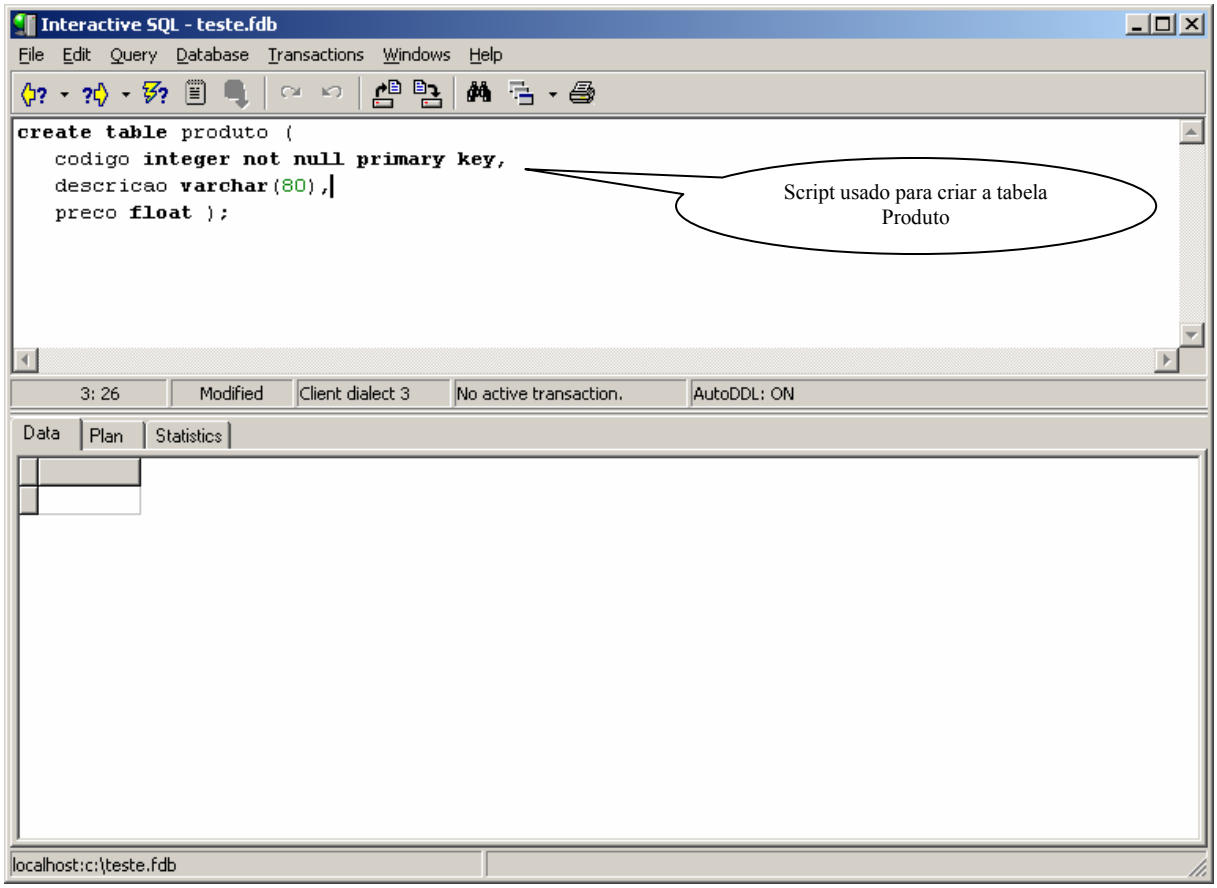

Figura 5 – Criação de tabela pelo IBConsole

Na Figura 6 é apresentada a tela de propriedades da tabela criada na figura 5. No IBConsole o acesso às propriedades é feito através do clique duplo na tabela desejada na sua tela principal.

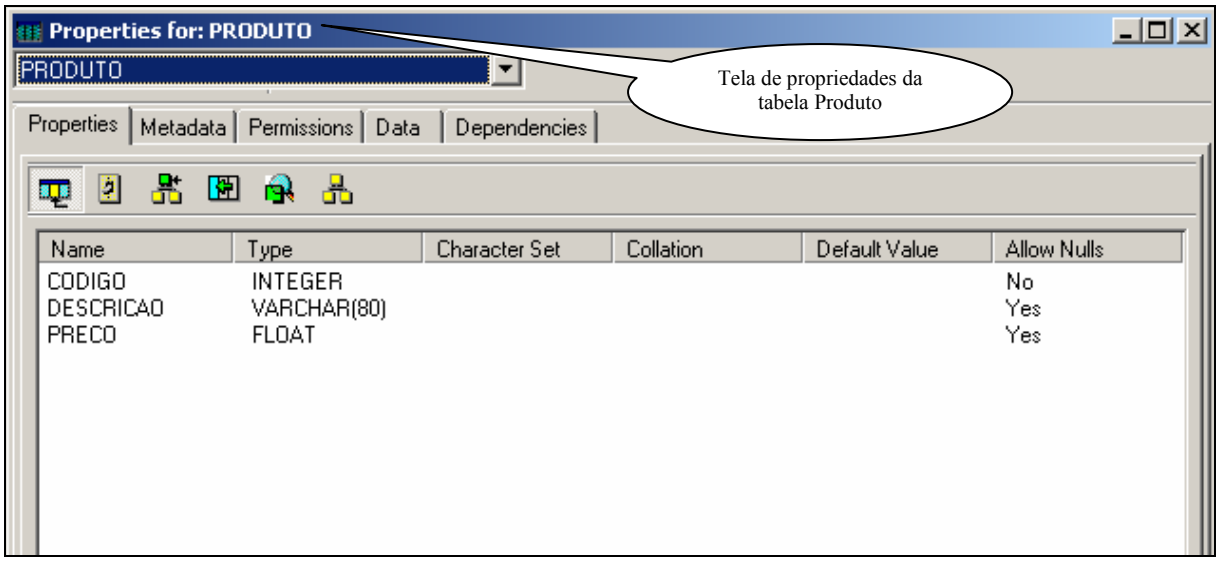

Figura 6 – Propriedades da tabela no IBConsole

A Figura 7 representa a inclusão de dados na tabela “PRODUTO” pela tela de propriedades, sendo que a mesma inclusão pode ser feita também pela submissão de execução de *scripts*.

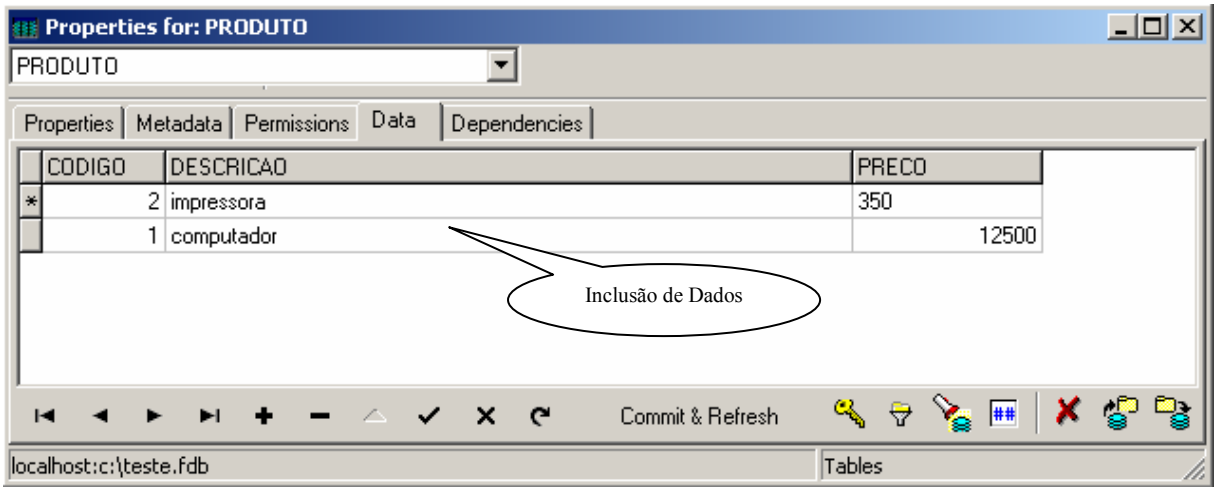

Figura 7 – Inclusão de dados na tabela Produto pelo IBConsole

3.2 IBEXPERT

O IBExpert foi desenvolvido pela empresa Inglesa HK Software, sendo considerado nos dias atuais por muitos programadores como a principal ferramenta de administração de bases de dados Interbase/Firebird. Entre os recursos que a tornam a principal ferramenta pode-se citar:

a) modelagem de dados (*Database Designer*);
b) gerador de documentação do banco de dados;
c) exportação de dados para Excel, RTF (Word), HTML, CSV, SYLK, DIF, TXT, LaTeX, XML.

A Figura 8 mostra a tela principal do IBExpert.

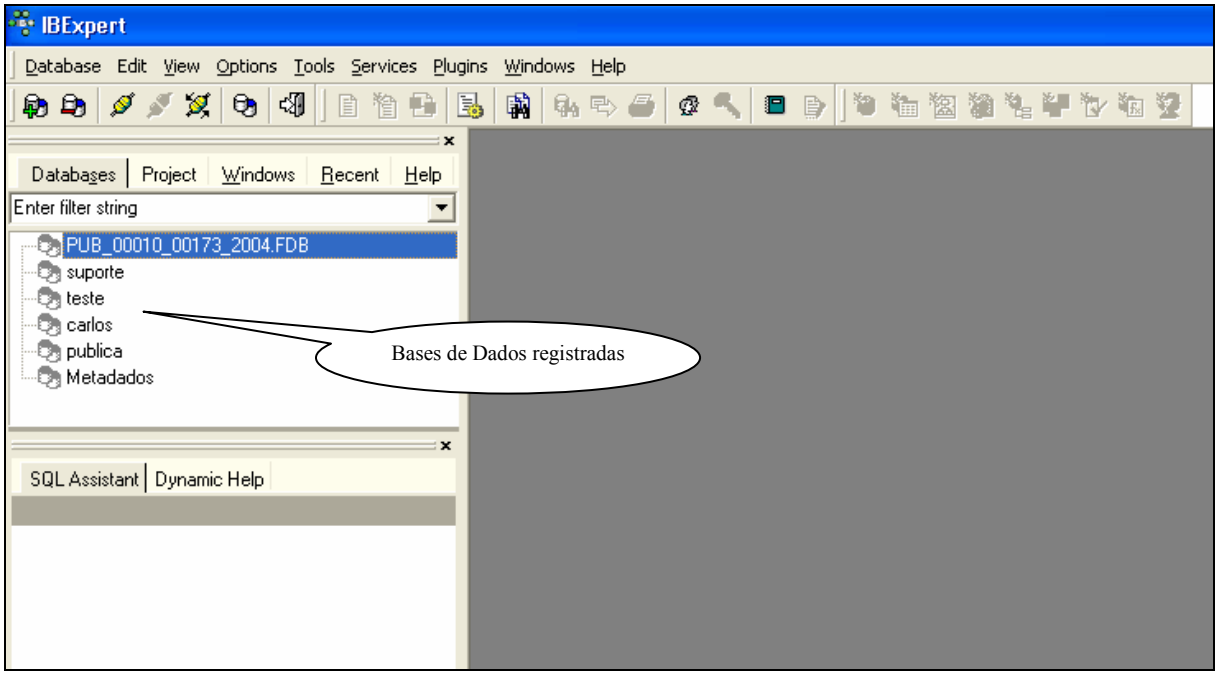

Figura 8 – Tela principal do IBExpert

A Figura 9 representa a criação de uma base de dados local com o nome de "TesteExpe.fdb" pelo IBExpert.

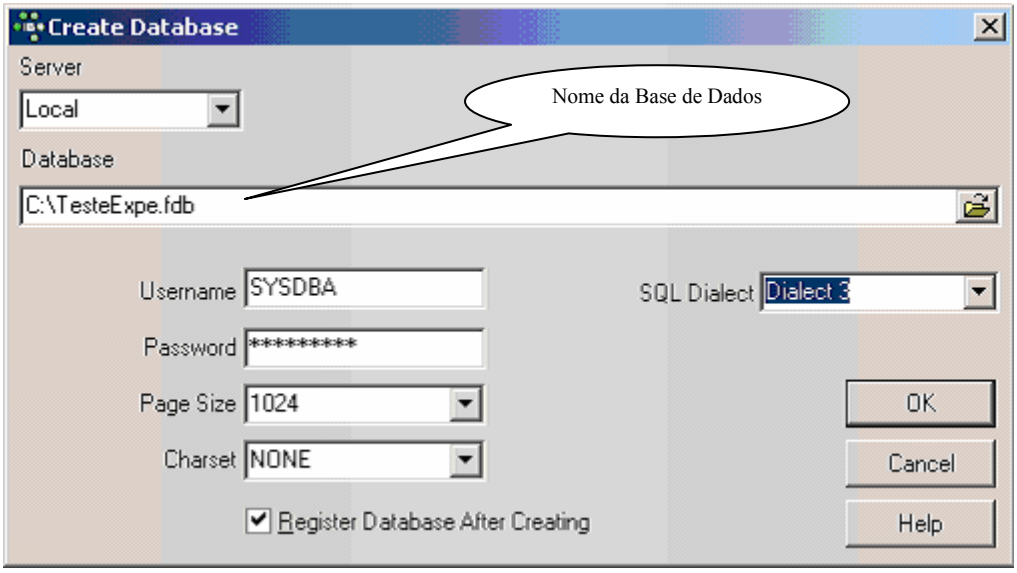

Figura 9 – Criação da base de dados pelo IBExpert

Na Figura 10 é demonstrada a criação de uma tabela utilizando o recurso gráfico do IBExpert, este recurso não está disponível no IBConsole.

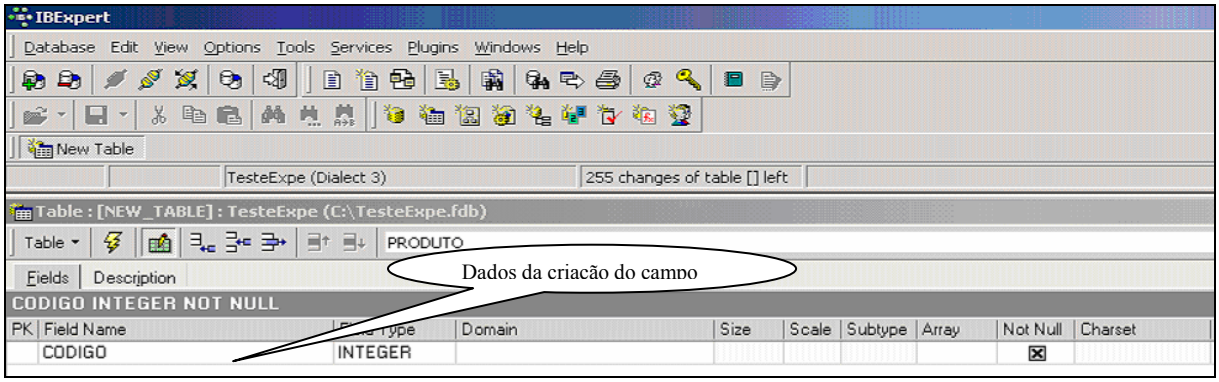

Figura 10 – Criação de tabela pelo IBExpert

O acesso as propriedades no IBExpert é direto pelo menu *Database Explorer* como pode ser visto na Figura 11. Estando disponível na tela de propriedades da tabela as estruturas dos campos, dados, metadados entre outros recursos.

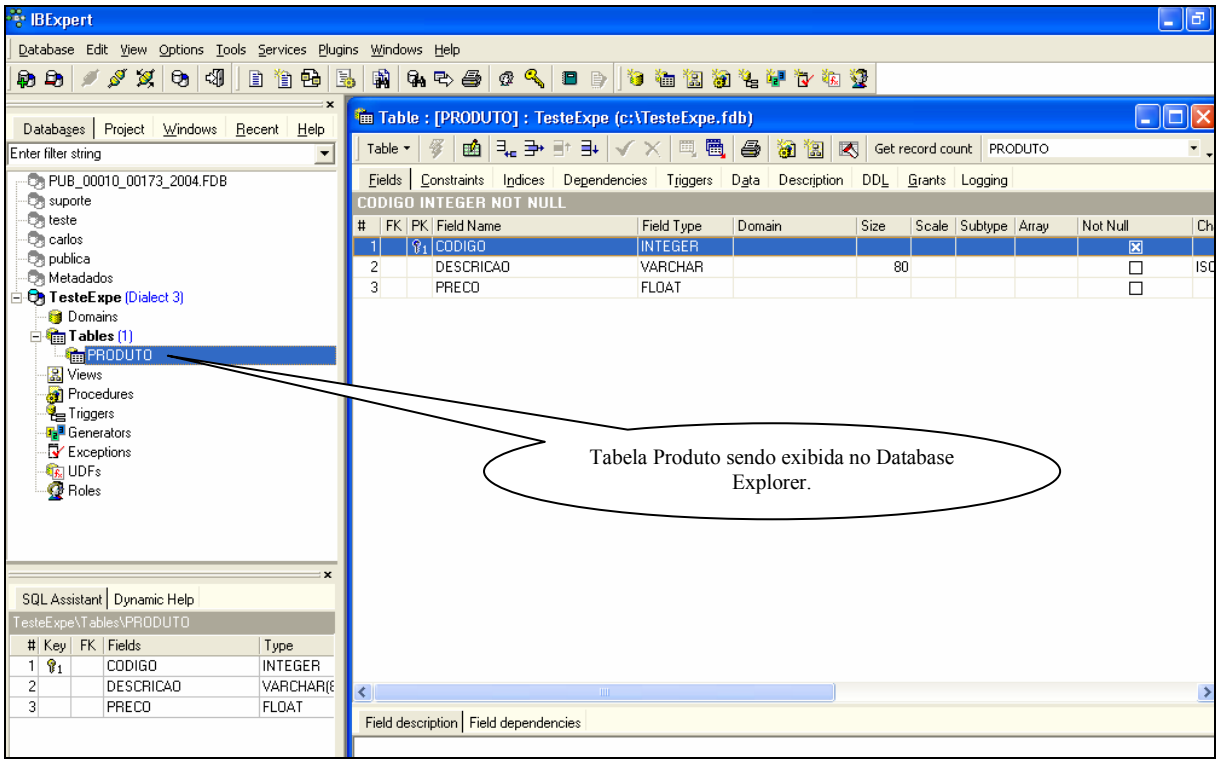

Figura 11 – Propriedades da tabela no IBExpert

O IBExpert, diferentemente do IBConsole, possui mais recursos além de sua interface ser mais agradável e de ter uma boa navegabilidade.

4 DESENVOLVIMENTO DO TRABALHO

A idéia para desenvolver esta ferramenta surgiu nas aulas de banco de dados, onde se verificou a falta de um recurso didático mais tangível sobre SGBDs *Open Source*, em especial o SGBD Firebird.

O desenvolvimento do trabalho descreve todo o processo de construção da ferramenta, iniciando pela especificação, implementação e apresentação da mesma.

O produto deste trabalho tem seu nome batizado de FBConsole.

4.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os principais requisitos do FBConsole são:

a) facilitar a visualização de bases de dados e tabelas;
b) permitir que o usuário execute comandos no formato SQL;
c) manter as propriedades de uma base de dados;
d) manter as propriedades de uma tabela;
e) manter dados de uma tabela;
f) permitir múltiplas conexões com o SGBD.

Com seu principal foco voltado para os dados, com o FBConsole o usuário (administrador) terá acesso a dados de uma tabela com maior facilidade.

4.2 ESPECIFICAÇÃO

A especificação é apresentada através do diagrama de casos de uso, de classes e de seqüência.

A Figura 12 representa os casos de uso que fazem parte da ferramenta onde se definem as funções do sistema. Os diagramas de casos de uso foram especificados utilizando a ferramenta Rational Rose.

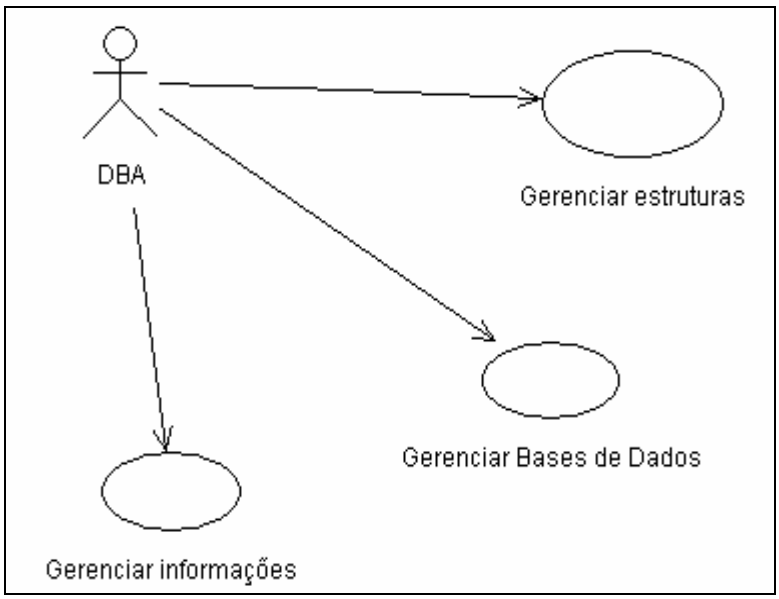

Figura 12 – Diagrama de casos de uso

A descrição de cada caso de uso e o nome dos casos é apresentado na Tabela3.

Tabela 3 – Descrição dos casos de uso

Caso de Uso	**Ator**	**Descrição**
Gerenciar estruturas	DBA	DBA manipula a estrutura da tabela, podendo criar, alterar ou apagar tabelas
Gerenciar Bases de Dados	DBA	DBA cria ou apaga as bases de dados existentes no SGBD
Gerenciar informações	DBA	Permite o gerenciamento da parte de dados das tabelas

Na Figura 13 observa-se através da representação no diagrama de classes a estrutura do sistema, bem como seus metadados e atributos.

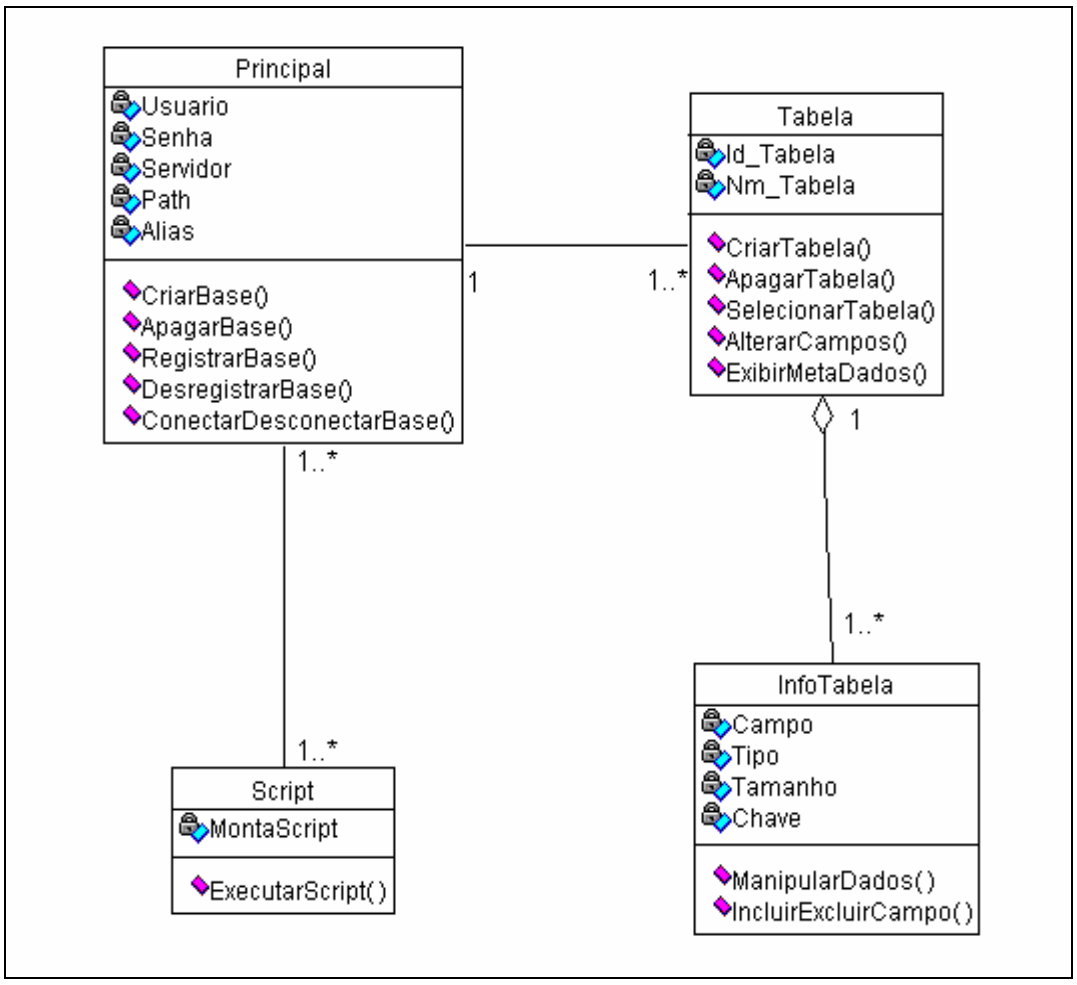

Figura 13 – Diagrama de classes

A classe Principal é a classe a qual toda a ferramenta está envolvida, e ao se executar a ferramenta é ela que é iniciada. Para se submeter um *script* dentro da ferramenta é utilizado a classe *Script*. A classe Tabela é responsável pela criação das tabelas, tendo uma agregação com a classe InfoTabela a qual permite a manutenção, inclusão e exclusão de novos campos nas tabelas do SGBD.

Na Figura 14 é especificado através de um diagrama de seqüência, o processo de criação de uma tabela e a sua futura inclusão de campos. Para este processo, o DBA inicialmente escolhe uma base de dados existente para se conectar. Em seguida o DBA terá acesso às propriedades da base de dados conectada, onde então será submetido a criação de uma tabela. Ao identificar uma tabela é verificado se a mesma já está incluída no banco de dados. Se ainda não estiver, inclui-se a tabela no banco de dados, e em seguida é permitida a manipulação dos campos dentro da tabela criada.

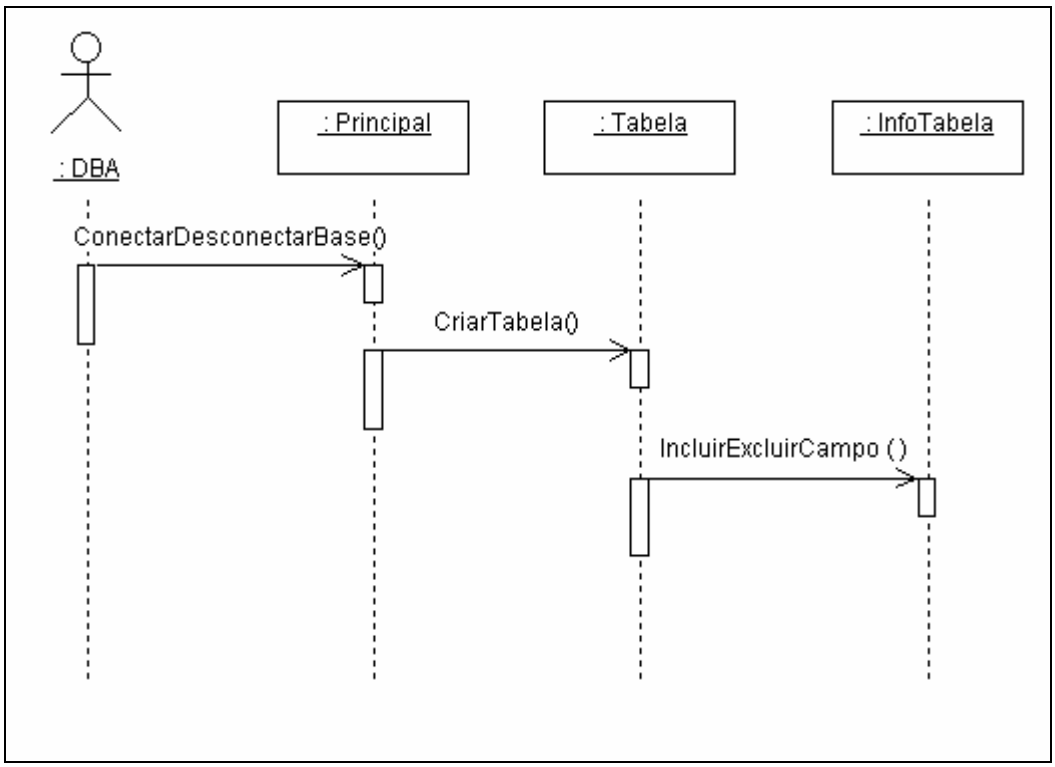

Figura 14 – Diagrama de seqüência – Gerenciar estruturas

A Figura 15 apresenta o processo de criação de uma base de dados.

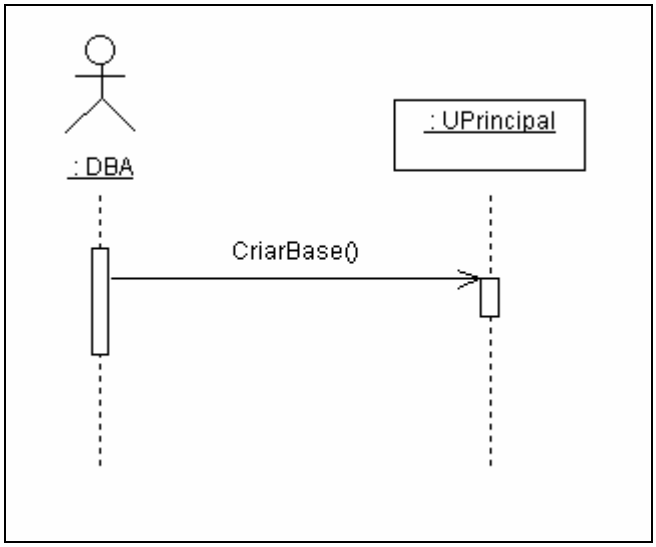

Figura 15 – Diagrama de seqüência – Gerencia Bases de Dados

Na Figura 16 é demonstrado o processo de manipulação de dados em tabela já existente.

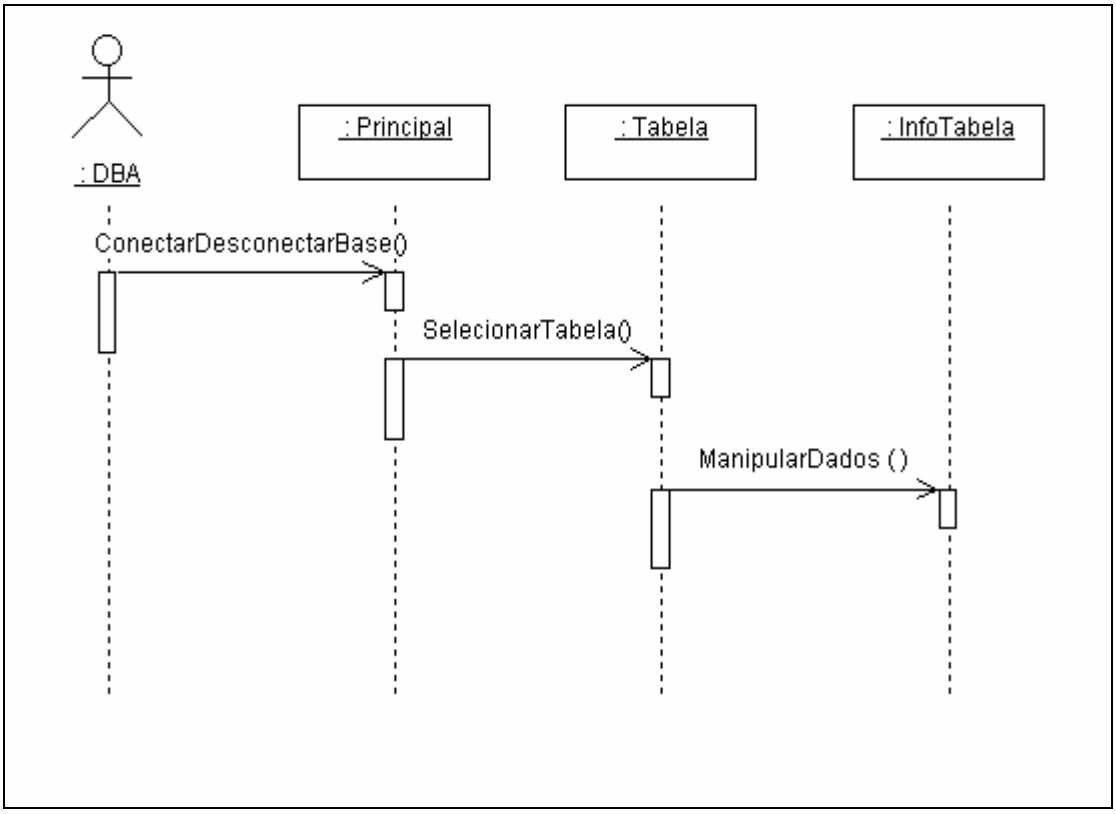

Figura 16 – Diagrama de seqüência – Gerenciar informações

4.3 IMPLEMENTAÇÃO

A seguir são apresentadas as técnicas e ferramentas utilizadas bem como a operacionalidade da implementação, onde são demonstradas as telas da ferramenta, bem como as características de cada uma delas.

4.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

A ferramenta foi desenvolvida utilizando o ambiente de desenvolvimento Borland Delphi 5 e seus componentes para a interação com o banco de dados e para a sua manipulação.

Para a comunicação do Delphi com o SGBD Firebird foram utilizados os componentes do pacote IBObjects de empresa IBObjects.

O pacote IBObjects de componentes é um dos mais abrangentes. Inclui além dos componentes de acesso aos dados componentes de edição de dados desenvolvidos especialmente para serem utilizados com o InterbaseF irebird, aproveitando ao máximo os recursos que o banco oferece. A ferramenta utilizou-se das duas ramificações que o IBObjects

oferece; sendo a primeira os componentes proprietários que devem ser utilizados com os componentes de edição de dados também próprios do IBObjects e a segunda que é composta por componentes de acesso derivados do TDataset e portanto compatíveis com qualquer componente de edição de dados do Delphi e de terceiros (*Edits*, *Grids*, etc...).

O IBObjects possui uma licença *Trustware* que permite que o desenvolvedor obtenha um registro temporário gratuíto até que obtenha retorno financeiro com o IBObjects e assim possa registrá-lo definitivamente. A Figura 17 demonstra os componentes IBObjects.

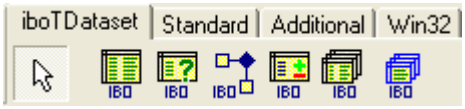

Figura 17 – Componentes IBObjects

Além do pacote IBObjects foi utilizado o componente *MemoryTable*(componente *Open Source* da empresa RxLib que pode ser encontrado no endereço http://www.clubedelphi.com.br/compo/detalhes.asp?id=153), que cria uma tabela na memória, utilizado para controlar as bases de dados registradas.

Como a ferramenta é um software de gerenciamento de um SGBD Firebird, para seu funcionamento será necessário a instalação de SGBD Firebird a partir da versão 1.0

Após a execução do aplicativo *Firebird Server Control* a ferramenta já estará em plenas condições de funcionamento. Com sua inicialização concluída a classe principal irá carregar o arquivo “fbconsole.ini” localizado no mesmo local do arquivo executável da ferramenta, sendo o mesmo responsável pela recuperação das últimas bases de dados utilizadas pela ferramenta. No Quadro 1 é demonstrada a abertura e leitura do arquivo “fbconsole.ini” e a inclusão dos dados contidos neste arquivo, assim como o caminho da base de dados e o nome do *alias* entre outros no componente *memory table* denominado de mtb_bases. O mtb_bases será responsável por controlar as bases de dados registradas na ferramenta. Após alimentar o mtb_bases será criado um nodo no componente do tipo *TreeView* chamado de *Tree_Menu*, que formará uma árvore com as bases de dados registradas. Conforme Cantú (2003, p. 156), o TreeView tem uma interface de usuário flexível e poderosa.

O controle para a alimentação do arquivo “fbconsole.ini” é feito quando a ferramenta tem o seu fechamento. Neste ponto é alimentado o arquivo com as informações contidas na mtb_bases como pode ser visto no Quadro 2.

```
mtb_Bases.open;
if fileExists(extractFilePath(application.ExeName)+ 'fconsole.ini')then
begin
  AssignFile(texto,extractFilePath(application.ExeName)+ 'fconsole.ini');
  Reset(texto); //abre o arquivo para leitura;
  While not eof(texto) do
  begin
    Readln(texto,linha); //le do arquivo e desce uma linha. O conteúdo lido é transferido para a variável linha
    mtb_Bases.insert;
    contador := Pos('#',linha);
    mtb_BasesNomeAlias.value := trim(copy(linha,1,contador-1));
    delete(linha,1,contador);
    contador := Pos('#',linha);
    mtb_BasesNomeServidor.value := trim(copy(linha,1,contador-1));
    delete(linha,1,contador);
    contador := Pos('#',linha);
    mtb_BasesPath.value := trim(copy(linha,1,contador-1));
    delete(linha,1,contador);
    contador := Pos('#',linha);
    mtb_BasesLogin.value := trim(copy(linha,1,contador-1));
    delete(linha,1,contador);
    mtb_BasesPassWord.value := trim(linha);
    mtb_Bases.post;
    fm_principal.Tree_Menu.items.addobject(nil,mtb_BasesNomeAlias.value,TIBODatabase.Create(Application));
  End;
  Closefile(texto);
```

Quadro 1 – Abertura do arquivo “fbconsole.ini”

```
Var
  Texto :TextFile;
begin
    if not(mtb_bases.isempty) then
    begin
      AssignFile(texto,extractFilePath(application.ExeName)+ 'fconsole.ini');
      Rewrite(texto); //abre o arquivo para escrita
      mtb_bases.first;
      while not(mtb_bases.eof) do
      begin
         //escreve no arquivo e desce uma linha
        Writeln(texto,mtb_BasesNomeAlias.text+'#'+mtb_BasesNomeServidor.text+'#'+
                      mtb_BasesPath.text+'#'+mtb_BasesLogin.text+'#'+mtb_BasesPassWord.text);
        mtb_bases.next;
      end;
      Closefile(texto);
    end
    else
    begin
      DeleteFile(extractFilePath(application.ExeName)+ 'fconsole.ini');
    end;
end;
```

Quadro 2 – Criação do arquivo “fbconsole.ini”

A Figura 18 demonstra o conteúdo do arquivo “fbconsole.ini”.

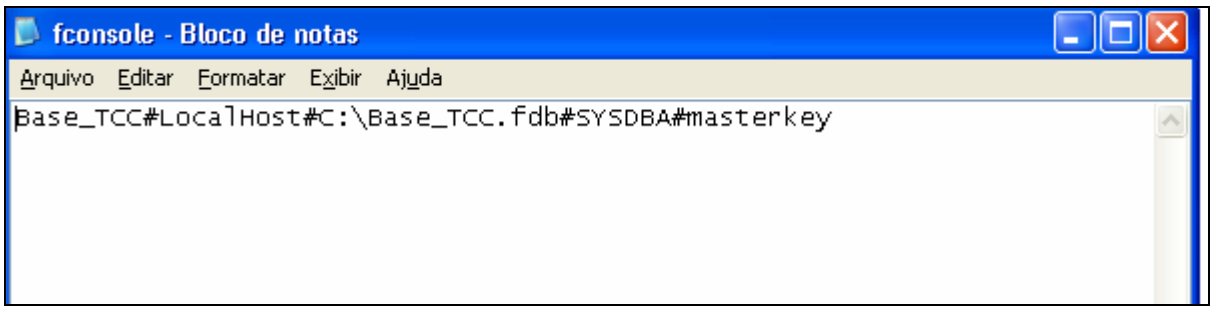

Figura 18 – Conteúdo do arquivo "fbconsole.ini"

Caso a execução da ferramenta se dê pela primeira vez, há a necessidade do registro de uma base de dados para sua plena funcionalidade. No Quadro 3 é demonstrado o método RegistrarBase onde são passados os atributos nome do servidor, local do base de dados, usuário, senha e nome do *alias*. Além destes atributos há uma verificação para que não haja mais de uma base de dados registrada com o mesmo *alias*, e após esta verificação, é alimentada na mtb_bases a base de dados correspondente, sendo a mesma adicionada em um novo nodo no componente *Tree_Menu* com o nome do *alias* de registro.

```
//se nao tiver vazio verifica se o Nome do Alias ja nao existe
fm_principal.mtb_Bases.first;
while not(fm_principal.mtb_Bases.eof) do
begin
  if fm_principal.mtb_BasesNomeAlias.value = NomeAlias.text then
  begin
    showmessage('Já existe uma base de dados registrada com este Nome de alias.');
    exit;
  end;
  fm_principal.mtb_Bases.next;
end;
fm_principal.mtb_Bases.insert;
fm_principal.mtb_BasesNomeAlias.value := NomeAlias.text;
if RB_Local.Checked = true then
  fm_principal.mtb_BasesNomeServidor.value := 'LocalHost'
else
  fm_principal.mtb_BasesNomeServidor.value := NomeServidor.text;
fm_principal.mtb_BasesPath.value := Database.Text;
fm_principal.mtb_BasesLogin.value :=  Login.Text;
fm_principal.mtb_BasesPassWord.value := Password.Text;
fm_principal.mtb_Bases.post;
```

Quadro 3 – Registro de base de dados

O registro de base de dados só é realizado através da existência de uma base de dados Interbase/Firebird criada, sendo assim, para a criação de uma base de dados será necessário a execução do método CriarBase dentro da classe principal, que depende basicamente dos atributos local de criação, usuário e senha onde a ferramenta irá utilizar o método já existente do componente *TIBODatabase* chamado de *CreateDatabase* para a criação da mesma. Uma

parte da implementação da criação de uma base de dados para o SGBD Firebird é demonstrada no Quadro 4.

```
TIBODatabase(Tree_Menu.Selected.Data).server := NomeServidor.text;
TIBODatabase(Tree_Menu.Selected.Data).Path   := Database.Text;
TIBODatabase(Tree_Menu.Selected.Data).Username := Login.Text;
TIBODatabase(Tree_Menu.Selected.Data).Password  :=Password.Text;
TIBODatabase(Tree_Menu.Selected.Data).SQLDialect := 3;
TIBODatabase(Tree_Menu.Selected.Data).CreateDatabase();
ButtonCria.Visible := false;
Pn_CriaBase.Visible := false;
```

Quadro 4 – Criação de uma base de dados

Após a criação do registro da base de dados, ou através de uma já existente poderá ser usado o método de ConectarDesconectarBase, que é utilizado para conectar ou mesmo desconectar uma base de dados, deixando ela assim apta a manutenção das tabelas. Quando é feita a conexão de uma base de dados será adicionado um nodo no componente Tree_Menu chamado de Tabelas sendo o mesmo responsável por exibí-las. No Quadro 5 é demonstrado onde a base de dados selecionada é conectada ou mesmo desconectada.

```
try
  if (TIBODatabase(xNodoDb.Data).Connected) and
     (pedido = 1) then
  begin
    showmessage('Base já está conectada.');
    exit;
  end;
  if pedido = 1 then
  begin
    TIBODatabase(xNodoDb.Data).server := mtb_BasesNomeServidor.value;
    TIBODatabase(xNodoDb.Data).Path   := mtb_BasesPath.value;
    TIBODatabase(xNodoDb.Data).Username := mtb_BasesLogin.value;
    TIBODatabase(xNodoDb.Data).Password  := mtb_BasesPassWord.value;
    TIBODatabase(xNodoDb.Data).Connected := true;
    Tree_Menu.Items.AddChild(xNodoDb,'Tabelas');
    sb_Principal.Panels[2].Text := TIBODatabase(xNodoDb.Data).Path;
  end
  else
  begin
    TIBODatabase(xNodoDb.data).Connected := false;
    if xNodoDb.level = 0 then
      xNodoDb.GetNext.Delete;
  end;
except
  showmessage('Houve Problema com a conexão a base de dados.');
end;
```

Quadro 5 – Conecta ou desconecta base de dados

Após a conexão de uma base de dados é permitido o início da manipulação de dados da mesma. Para que possa carregar as propriedades da estrutura física de uma tabela selecionada é utilizado o metadados do SGBD Firebird. É demonstrado no Quadro 6 uma parte da extração do metadados utilizando o recurso do SQL para obtenção das informações necessárias da tabela selecionada.

```
vQuery.Sql.Text := 'SELECT A.RDB$FIELD_NAME AS Campo,          '+
                   '       C.RDB$TYPE_NAME AS Tipo,            '+
                   '       B.RDB$FIELD_LENGTH AS xSize,        '+
                   '       A.RDB$NULL_FLAG as EHNOTNULL,       '+
                   '       A.RDB$COLLATION_ID As xCollate,     '+
                   '       RDB$FIELD_SUB_TYPE AS Sub,          '+
                   '       B.RDB$CHARACTER_SET_ID as CharSet,  '+
                   '       B.RDB$FIELD_SCALE as xDecimal,      '+
                   '       A.RDB$DESCRIPTION as Descricao      '+
                   'FROM RDB$RELATION_FIELDS A                 '+
                   'LEFT JOIN RDB$FIELDS B ON A.RDB$FIELD_SOURCE=B.RDB$FIELD_NAME  '+
                   'LEFT JOIN RDB$TYPES  C ON B.RDB$FIELD_TYPE  =C.RDB$TYPE        '+
                   'WHERE A.RDB$RELATION_NAME = '+quotedstr(pTabela)+' AND '+
                   '    C.RDB$FIELD_NAME= '+quotedstr('RDB$FIELD_TYPE')+'          '+
                   'ORDER BY RDB$FIELD_POSITION                ';
vQuery.Open;
```

Quadro 6 – Metadados da tabela existente

A extração do *script* de execução da criação da tabela é feito através do componente TIB_Metadata onde o mesmo possui um método chamado ExtractFull que retorna o *script*, como é demonstrado a sua implementação no Quadro 7.

```
Procedure TFm_Principal.GetScript(pBanco : TIBODatabase; pTabela : string; pMemoDDL : TMemo);
var
  Metadata: TIB_Metadata;
  tmpStrs: TStrings;
  idx, vstart : smallint;
begin
  try
    tmpStrs := TStringList.Create;
    Metadata := TIB_Metadata.create(Application);
    Metadata.IB_Connection := pBanco;
    Metadata.Ib_Transaction := pBanco.Ib_Transaction;
    TabelaAtual := pTabela;
    Metadata.OnFilterByName := FiltraTabela;
    Metadata.ShowProblems := false; // para depurar possiveis problemas ligar este flag
    Metadata.ExtractFull([etTable,etPrimaryKey,etUniqueKey,etIndex,etForeignKey,etConstraint],tmpStrs);
    pMemoDDL.Lines.add(tmpStrs.Text);
  finally
    tmpstrs.Free;
    Metadata.free;
  end;
end;
```

Quadro 7 – *Script* de criação da tabela

A criação de uma tabela no FBConsole pode ser realizada através da execução de *script* ou pelo recurso gráfico disponibilizado pela ferramenta. No Quadro 8 é demonstrado a criação de uma tabela onde é verificado a base de dados selecionada e se o nodo selecionado é o *alias* ou mesmo o nodo tabelas. Caso o nodo selecionado seja o tabelas irá utilizar o método *GetPrev* do componente *Tree_Menu* para retornar o nome do *alias* da base de dados.

```
// Executa Query
try
  if fm_principal.Tree_Menu.Selected.level = 0 then
    xnodo := fm_principal.Tree_Menu.Selected
  else
    xnodo := fm_principal.Tree_Menu.Selected.GetPrev;
  dm_Principal.qr_Principal.IB_Connection := TIBODatabase(xnodo.Data);
  if not dm_Principal.qr_Principal.IB_Connection.DefaultTransaction.InTransaction then
    dm_Principal.qr_Principal.IB_Connection.DefaultTransaction.StartTransaction;
  dm_Principal.qr_Principal.Prepared;
  dm_Principal.qr_Principal.ExecSQL;
  dm_Principal.qr_Principal.IB_Connection.DefaultTransaction.Commit;
  showmessage('Tabela criada com Sucesso');
  Lista.Items.Clear;
except
  showmessage('Não foi possivel criar a tabela');
  exit;
end;
```

Quadro 8 – Criar tabela

4.3.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Nesta seção serão apresentadas as principais telas da ferramenta que permitirão o melhor entendimento da mesma.

A Figura 19 apresenta a tela principal da ferramenta, que é dividida em quatro menus. O menu “Base Dados” é responsável pelas bases de dados utilizadas. No menu “Exibir” são utilizados os controles de exibição dos itens. O menu “Ferramenta” disponibiliza o recurso de execução de *scripts*. Por fim no menu ajuda se encontra a tela sobre o sistema e a opção para fazer o *download* do Firebird.

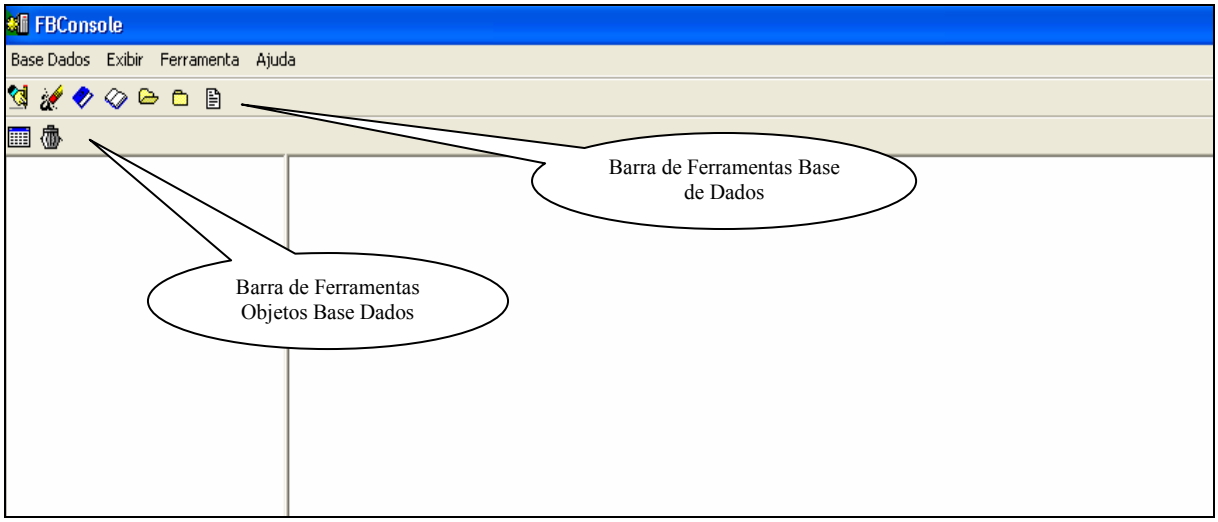

Figura 19 – Tela principal do FBConsole

A Figura 20 demonstra os itens que estão nas barras de ferramentas “base de dados” e “Objetos Base Dados”.

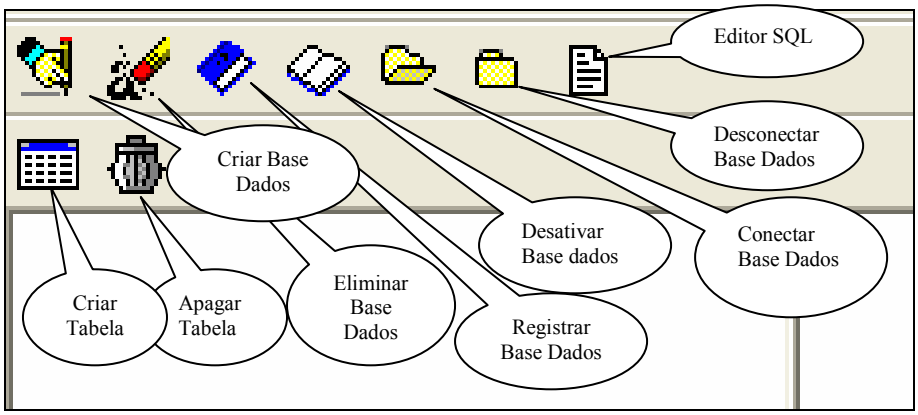

Figura 20 – Barra de Ferramentas do FBConsole

A Figura 21 demonstra a página acessada pelo Menu Ajuda, item baixar o Firebird, que tem como intuito possibilitar de maneira fácil e rápida a cópia e instalação do SGBD Firebird.

Figura 21 – Página para download do Firebird

Após a instalação do SGBD Firebird e da inicialização do *Firebird Server Control* a ferramenta está pronta para manuseio de bases de dados.

A Figura 22 demonstra a criação de uma base de dados local utilizando o nome de Base_Tcc.fdb onde o usuário será o “SYSDBA” e sua senha “masterkey”, ambos padrão em um SGBD Firebird. Além da criação da base de dados é permitida a ativação do recurso de “Registrar a Base após criar” onde é feito o registro da base de dados na ferramenta, possibilitando assim a mesma de ser utilizada pelo FBConcole. Este recurso de registro de bases de dados também está disponível no menu Base Dados na opção “Registrar Base Dados” conforme é demonstrado na Figura 23.

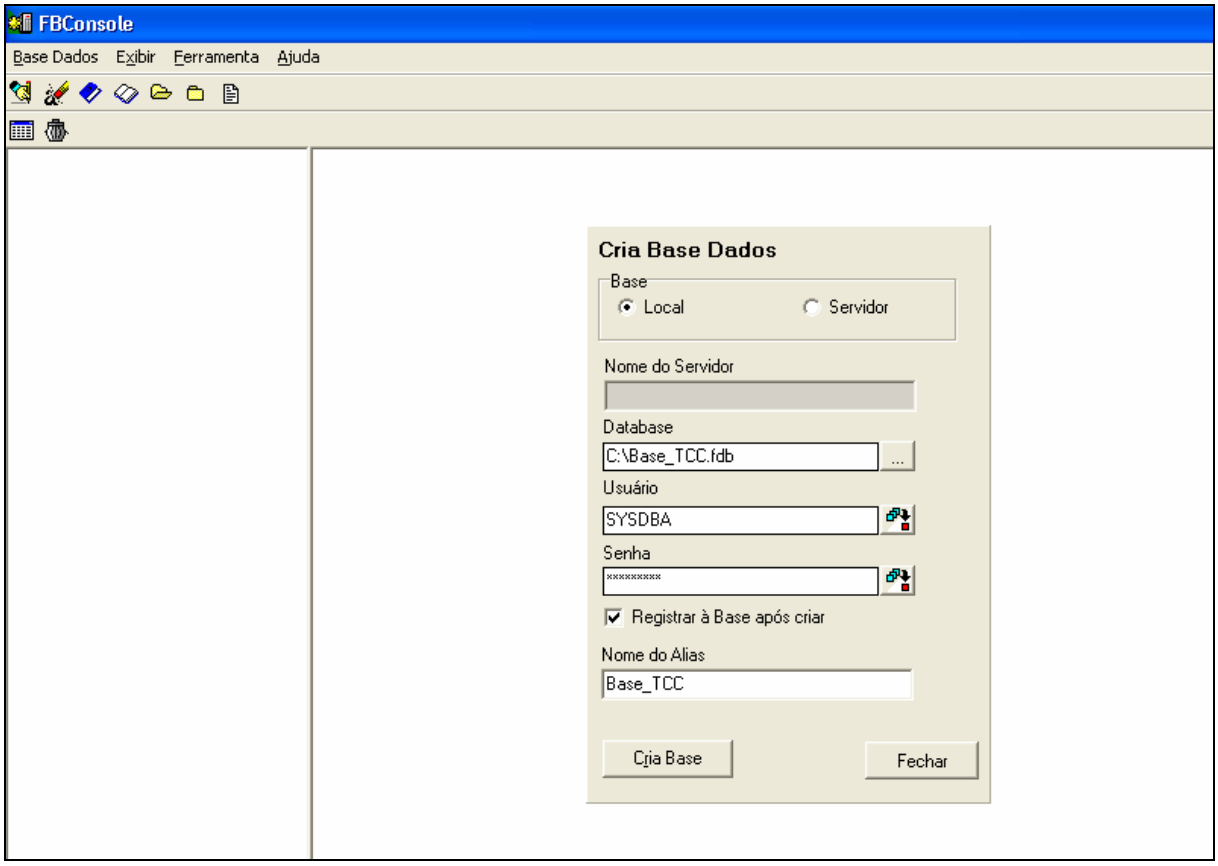

Figura 22 – Criação de uma base de dados

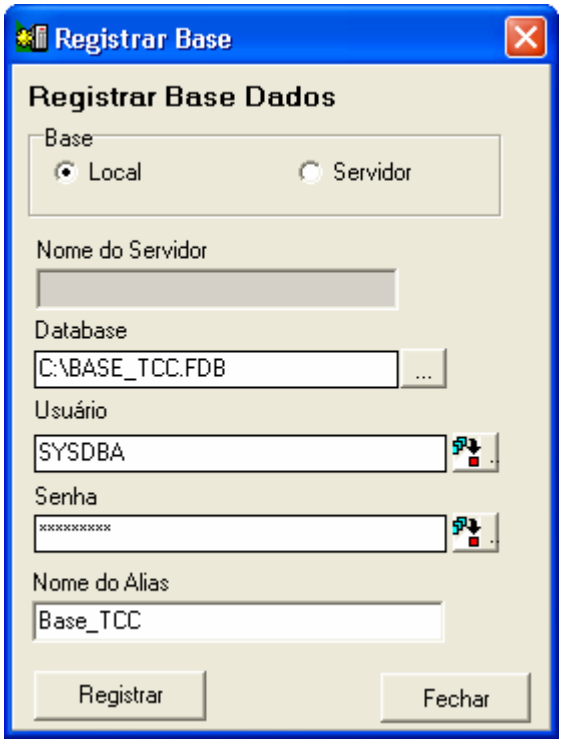

Figura 23 – Registro da base de dados “Base_TCC”

Após o registro da base de dados já é possível a criação de tabelas. A criação de uma tabela poderá ser feita pelo botão “Criar Tabela”. Esta opção é encontrada na barra de ferramentas “objetos Base Dados”. As barras de ferramentas têm sua exibição pelo menu “Exibir” – “Barra de Ferramentas”.

A Figura 24 demonstra a criação da tabela propriamente dito, onde é criada uma tabela chamada “produto” com os campos codigo, descricao e preço. Destes, o campo codigo foi definido como chave primária.

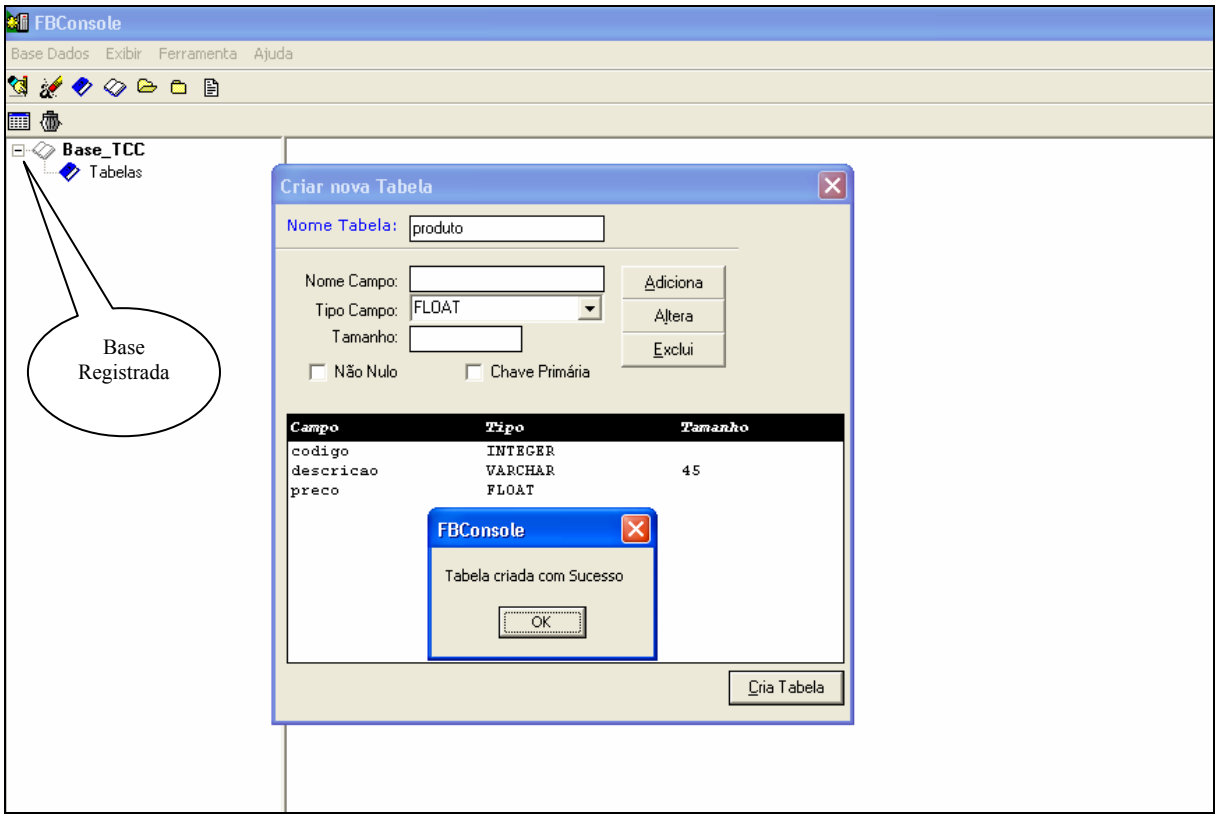

Figura 24 – Criação de uma tabela utilizando recurso gráfico

Além do recurso gráfico para criação da tabela a mesma poderá ser criada pelo execução de *script*, que se encontra no menu Ferramenta opção *Script.* A Figura 25 demonstra a criação de uma tabela chamada “Partes” pela submissão de um *script*.

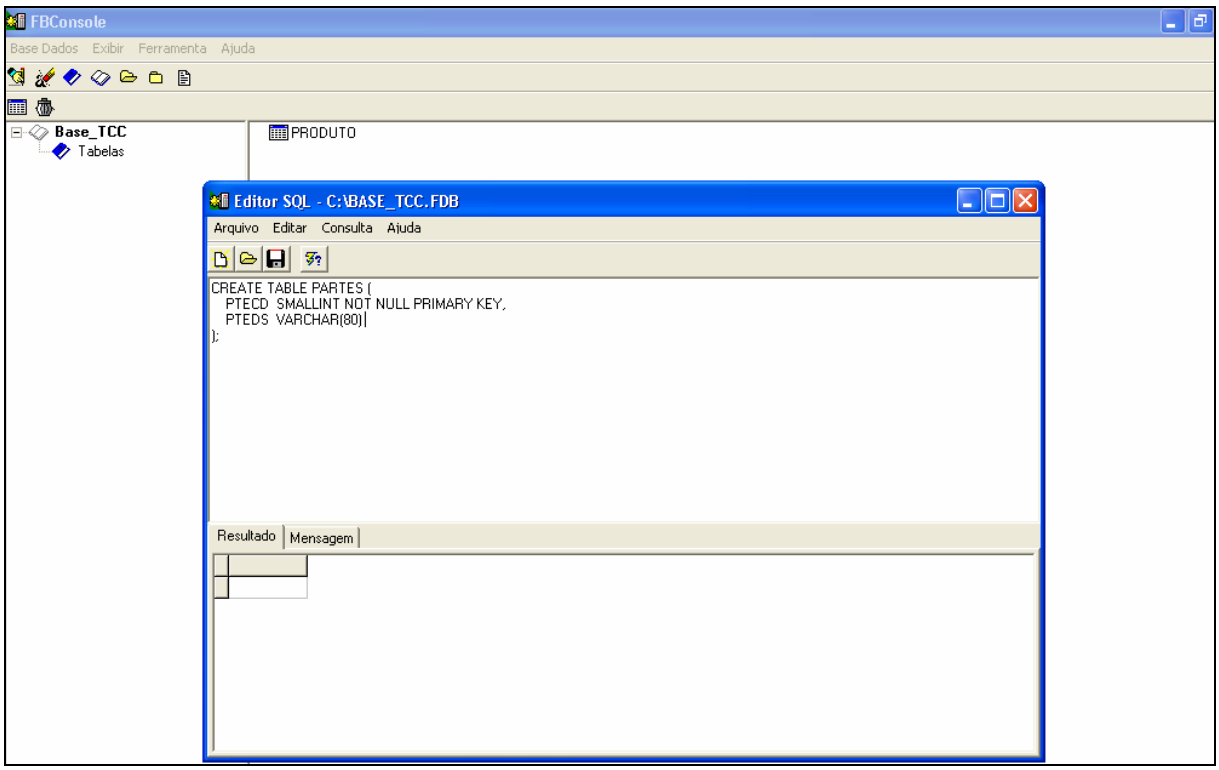

Figura 25 – Criação de uma tabela utilizando recurso de submissão de *script*

Após a criação da tabela a mesma estará disponível pelo duplo clique do mouse no nodo tabelas dentro do nodo principal "Base_TCC", conforme é demonstrado na Figura 26.

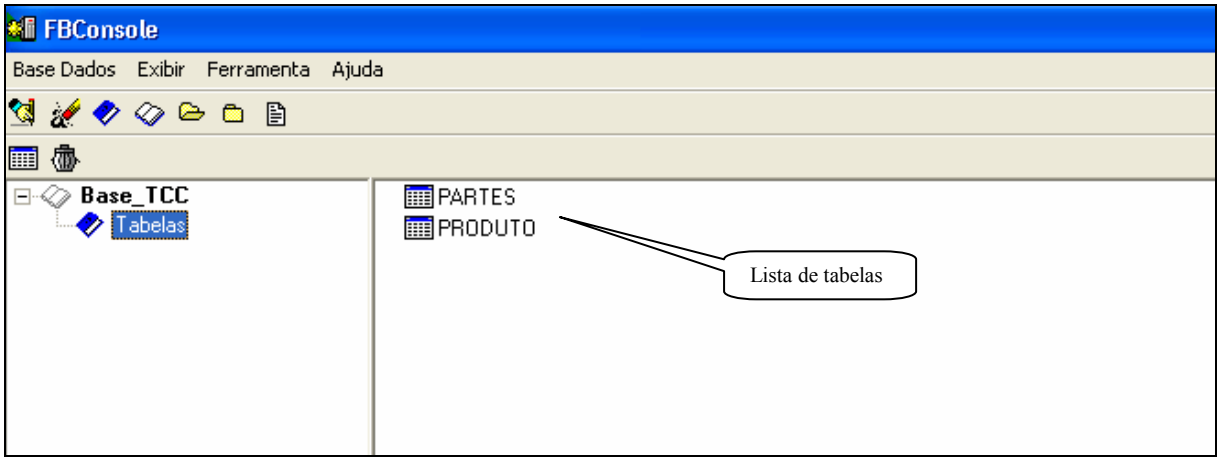

Figura 26 – Listagem de Tabelas

Para que seja possível a visualização das propriedades da tabela será necessário o duplo clique do mouse no nome da tabela especificada, neste caso "produto". A Figura 27 demonstra as propriedades da tabela sendo elas: colunas, índices, dados, metadados. A guia chamada "colunas" é responsável pela manutenção dos respectivos campos da tabela.

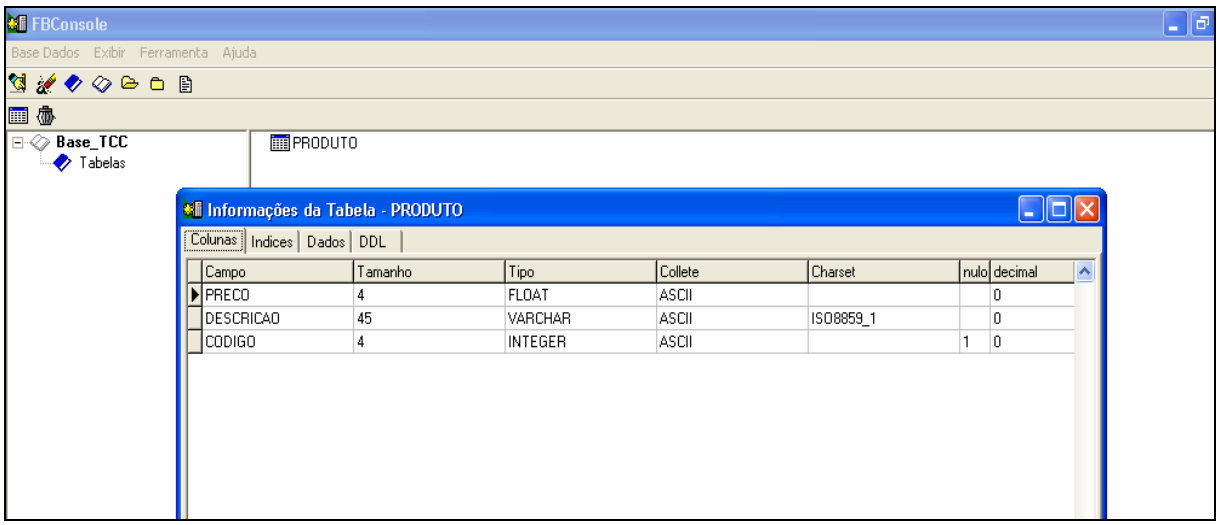

Figura 27 – Informações de uma tabela

A guia "índices" tem como objetivo mostrar os índices e restrições das tabelas, como é demonstrado na Figura 28.

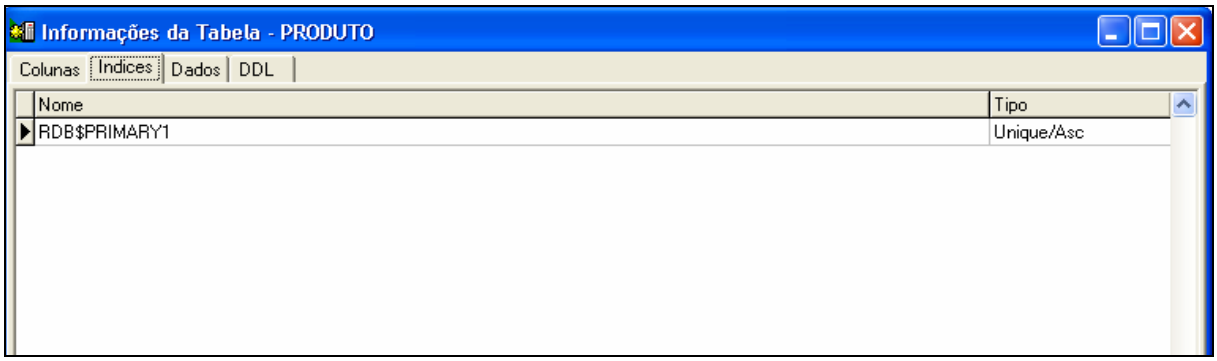

Figura 28 – Restrições da tabela

A inclusão de dados no FBConsole pode ser efetuada tanto pela submissão de *scripts* conforme é demonstrado na Figura 29 ou ainda pelo recurso gráfico como pode ser visto na Figura 30.

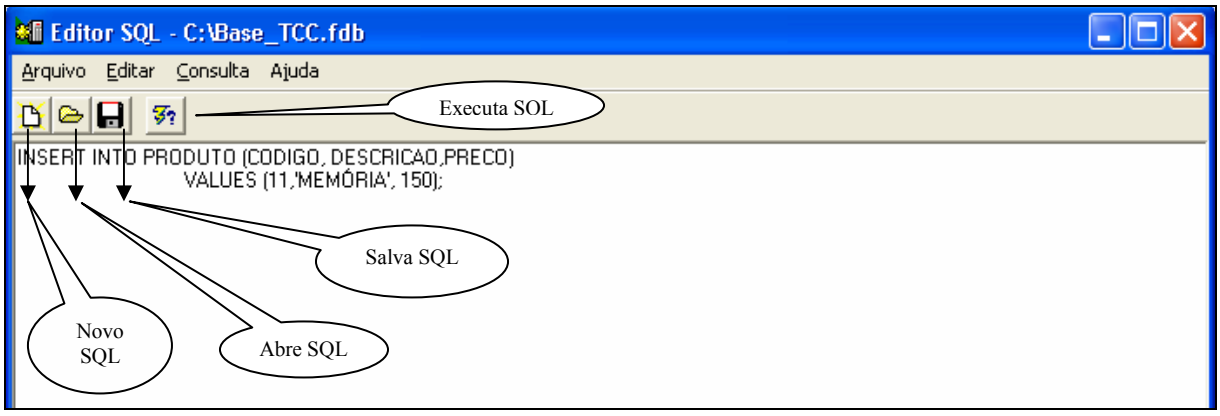

Figura 29 – Inserção de dados via *script*

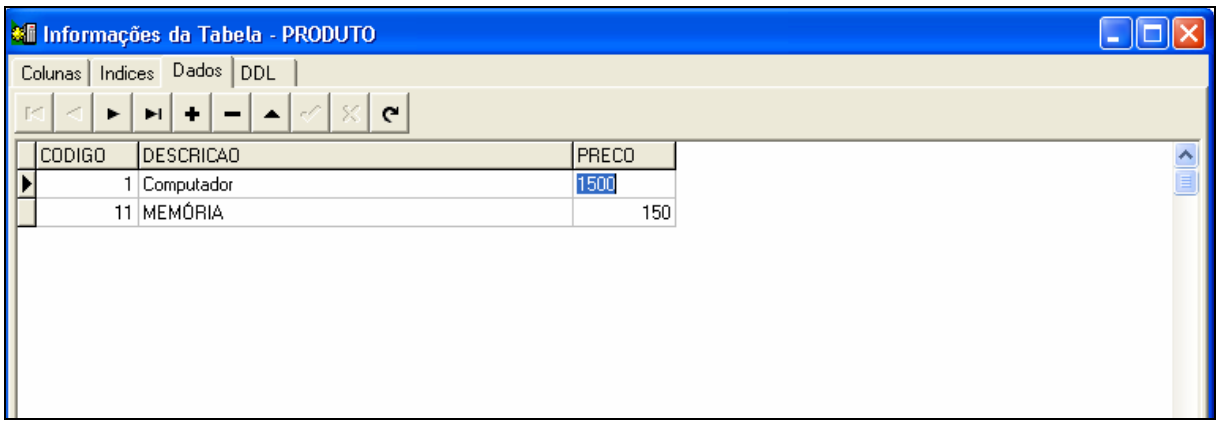

Figura 30 – Inserção de dados via recurso gráfico

A última guia demonstra os comandos para criação da tabela (DDL) conforme é demonstrado na Figura 31. A seqüência de comandos pode ser utilizada para a criação da mesma tabela em outra base de dados.

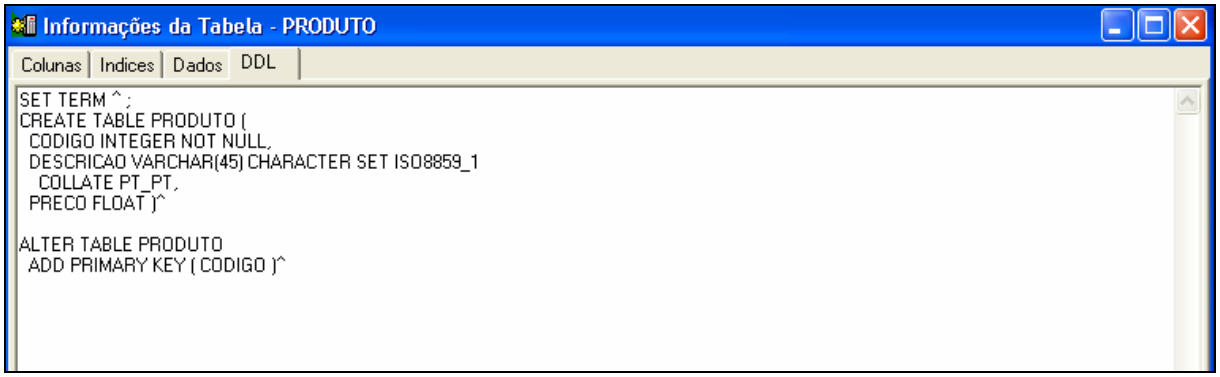

Figura 31 – DDL da tabela

4.4 RESULTADOS E DISCUSSÃO

Para o desenvolvimento da ferramenta FBConsole foram utilizados os componentes padrões do Delphi. Entre estes pode-se citar o *TreeView* como o que apresentou maior dificuldade para sua utilização, devido a escasses de material de referência, uma vez que a literatura sobre o ambiente Delphi não ilustra/demonstra claramente a utilização. Além dos componentes padrões foram adicionados os componentes *MemoryTable* e o *IBObjects*, os quais oferecem um bom desempenho.

Pelo fato do Firebird ser um SGBD novo no mercado houve muita dificuldade em encontrar as informações referentes ao seu metadados e posteriomente as informações técnicas.

O FBConsole em relação as ferramentas apresentadas nesta monografia tem como principais vantagens a criação de tabelas utilizando o recurso de *scripts* e gráfico (através de componentes), código fonte disponível para alterações e a sua distribuição gratuita para o SGBD Firebird.

5 CONCLUSÕES

A partir de um projeto desenvolvido no ambiente de programação Borland Delphi, é possível o gerenciamento de um SGBD Firebird.

Para a elaboração deste trabalho foram estudados diversos conceitos durante o seu desenvolvimento, como SGBDs, ferramentas de gerenciamentos, SQL e principalmente o metadados.

Foram feitas tentativas utilizando ferramentas Case através de engenharia reversa, para investigar as relações entre as tabelas do metadados. Todavia sem sucesso uma vez que o metadados do Firebird não apresenta um relacionamento explicito entre as tabelas, não sendo intuitivo e perceptível.

O estudo e desenvolvimento do trabalho foram de grande importância devido a escassez de referências sobre o tema, motivo causado por ser um SGBD relativamente novo no mercado, mas que vem crescendo devido as suas muitas aplicações.

Os objetivos propostos no início deste trabalho foram alcançados com êxito.

5.1 EXTENSÕES

Buscando aprimorar os resultados obtidos com a ferramenta, e um aprofundamento melhor sobre o tema, sugere se:

a) implementar um gerenciamento de *triggers*;
b) implementar um gerenciamento de tipos de dados;
c) implementar um gerenciamento de visões;
d) implementar um gerenciamento de linguagem procedural.
e) Implementar um gerenciamento de usuário.

REFERÊNCIAS BIBLIOGRÁFICAS

CANTU, Carlos H. **Firebase**, [S.l],[2002]. Disponível em: < http://www.firebase.com.br>. Acesso em: 10 abri. 2004.

CANTÚ, Marco. **Dominado o Delphi 7**: a bíblia. São Paulo: Pearson Education, 2003. 801 p.

DATE, C. J. **Introdução a sistemas de bancos de dados**. Rio de Janeiro: Campus, 2000. 803 p.

FREITAS, Alfredo Américo de. **Firebird.Br**, [S.l], [2002]. Disponível em: <http://www.firebird.com.br/index.php>. Acesso em: 19 set. 2003.

MECENAS, Ivan. **InterBase 6** : guia do desenvolvedor. Rio de Janeiro: Book Express, 2000. 156 p.

RODRIGUES, Anderson Haertel. **Apostila de Firebird 1.0**, [S.l],[2002]. Disponível em: <http://www.comunidade-Firebird.org/>. Acesso em: 15 set. 2003.

WEB Information. **Banco de dados**, [S.l], [2002]. Disponível em: http://www.webinformation.hpg.ig.com.br/computer_bancodedados.htm>. Acesso em: 18 set. 2003.