UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

FERRAMENTA DE CONTROLE E MANUTENÇÃO DE BASES DE DADOS

EDUARDO MENDES DE CÓRDOVA

BLUMENAU
2004

2004/1

EDUARDO MENDES DE CÓRDOVA

FERRAMENTA DE CONTROLE E MANUTENÇÃO DE BASES DE DADOS

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. Mauro Marcelo Mattos - Orientador

BLUMENAU
2004

2004/1

FERRAMENTA DE CONTROLE E MANUTENÇÃO DE BASES DE DADOS

Por

EDUARDO MENDES DE CÓRDOVA

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

Presidente:	Prof. Mauro Marcelo Mattos – Orientador, FURB
Membro:	Prof. Jomi Fred Hübner, FURB
Membro:	Prof. Alexander Roberto Valdameri, FURB

AGRADECIMENTOS

À Deus, por ter-me concebido cada momento de minha vida.

À minha família, na qual sempre pude contar com o auxílio nos momentos difíceis, e grandes alegrias nos bons momentos, especialmente aos meus pais.

Ao meu chefe, Aoron Beyer, pela compreensão e amizade. Aos meus colegas de trabalho, bons amigos que posso contar com a ajuda quando necessário.

À Juliana Donatz, que me fez descobrir a verdadeira amizade e cumplicidade entre duas pessoas, e como a vida é maravilhosa quando com o amor presente.

Ao meu orientador, Mauro Marcelo Mattos, pela sua paciência e auxílio. Por mim admirado pelo seu caráter e por ser um grande professor.

Aos meus grandes amigos, Nélvio Burati, Rangel Bordin, Ricardo Tomedi, Lúcio Rebello, Alexandre Strube e outros, que sempre estarão no meu coração e nunca serão esquecidos, e que espero sempre ter as suas companhias.

RESUMO

Este trabalho descreve a criação de uma ferramenta que trata da manipulação, visualização e gerenciamento de um esquema de dados lógico aplicado fisicamente em uma base de dados. Com o uso da ferramenta, é possível a um projetista com pouco ou nenhum conhecimento em SQL gerenciar a evolução de um esquema de dados, sendo os comandos SQL necessários para a manipulação das bases de dados gerados pela ferramenta. A ferramenta possui um sistema de controle de versões das bases utilizadas, possibilitando a geração de *scripts* definidos pelos projetistas contendo um conjunto de alterações a serem aplicadas a uma base e utilizados para realizar a evolução do esquema de dados de bases conforme a base de referência.

Palavras chaves: Ferramenta de suporte, geração de esquema de dados, controle de versões de bases de dados.

ABSTRACT

This work describes the creation of a tool that handle the manipulation, visualization and management of a logical data scheme fisically aplyed in databases. By using the tool, it's possible for a designer with none or little knowledgement in SQL to manage and implant the data scheme evolution, being the SQL commands necessaries for the database manipulation generated by the tool. The tool has a version control system for manipulated databases, thus enabling the script generation defined by the designers that contains a whole alterations to be aplyed in a database and used to accomplish the database scheme evolution according the reference database.

Keywords: Suport tool, database scheme generation, database version control.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

As pesquisas na área de bases de dados sempre estiveram sincronizadas com as necessidades dos usuários (SILBERSCHATZ, 1996). Os objetivos e resultados destas pesquisas, antigamente, eram voltadas somente para a aplicação genérica (denominada "convencional"), todavia com o crescimento e diversificação do mercado de *software* para usuários e conseqüentemente aumento da complexidade das aplicações, tornaram-se também necessárias as pesquisas para o desenvolvimento de Sistema de Gerenciamento de Bases de Dados (SGBD's) mais específicos (SILBERSCHATZ, 1996).

O reflexo das necessidades do mercado pode ser cronologicamente visualizado através dos diversos modelos de dados desenvolvidos, desde modelos baseados em registros (rede, hierárquico e relacional) até os modelos orientados a objetos.

Constantemente as empresa têm reavaliado suas estruturas e processos de trabalho de modo a tornarem-se mais eficientes na qualidade de seus produtos ou serviços. Em empresas e universidades, a preocupação está em acelerar os processos criativos para a obtenção de resultados reais, fundamentados e relacionados com os problemas atuais.

Entre os processos de trabalho, o nível comercial, está o desenvolvimento de produtos ou a apresentação de resultados que ofereçam inovações para satisfazer clientes e assim almejar conquistas no mercado consumidor, e ao nível científico, é apresentar soluções atuais para os problemas existentes. Isto motiva a pesquisa e o nascimento de diversas ferramentas para apoio a projetos das mais distintas áreas.

O apoio ao grupo de projetistas, através de ferramentas, tornou-se indispensável pois oferece agilidade, segurança, e flexibilidade de trabalho. A utilização e o compartilhamento de informações exige a utilização de um ambiente gerenciador, onde as informações são armazenadas em bases de dados.

Um problema encontrado no desenvolvimento de aplicações que utilizam banco de dados, é que durante todo o seu ciclo de vida, há a necessidade de realizar modificações na base para suprir as necessidades do mercado, resultando na criação de novas versões de um sistema, gerando uma grande necessidade de possuir um controle de versões e alterações das bases de forma segura e confiável.

O presente projeto descreve o desenvolvimento de uma ferramenta que possibilita ao projetista criar, modificar e excluir entidades em uma base de dados relacional de forma simplificada e sem a necessidade de elaboração de comandos SQL. Através da utilização de formulários, é possível informar as definições da estrutura das tabelas, campos e índices, como nome, tipo, tamanho, relacionamentos entre outros. A ferramenta interpreta as definições informadas, gerando o código SQL para efetivar as alterações na base, tornando abstrato ao usuário os comandos para a manipulação da estrutura.

As alterações da estrutura da base são realizadas de forma direta, aplicadas na estrutura física da base a cada modificação. A ferramenta armazena em *log* cada alteração realizada, gerando assim um histórico da evolução do esquema de dados da base. Quando uma atualização de base for realizada, é gerado um *script* de exportação das alterações implementadas na base de desenvolvimento, podendo este ser específico para cada base de produção, proporcionando uma forma segura e eficaz de atualização.

Este projeto está inserido no contexto de uma *software house* que comercializa produtos de software para seus clientes. Neste sentido, a denominação clientes utilizada neste trabalho faz referência aos clientes da *software house* que utilizam os sistemas por esta desenvolvidos, e denominados de projetistas os usuários da *software house* que utilizam a ferramenta proposta neste trabalho.

Serão referenciadas como base de desenvolvimento a base utilizada pela *software house* onde são implementadas as definições do esquema de dados, e bases de produção como as bases utilizadas pelos clientes da *software house* para o armazenamento dos dados de seus sistemas.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver uma ferramenta de manutenção e alteração de bases de dados, onde é possível ter o controle de versão da base de dados de desenvolvimento e das bases de produção de forma eficaz.

Os objetivos específicos do trabalho são:

- Permitir a criação, alteração e visualização da estrutura de bases de dados sem a necessidade do conhecimento da linguagem SQL;
- Gerenciar a atualização de bases, gerando *scripts* de exportação de alterações da base, e a atualização na base cliente (base de produção);
- Permitir a execução de comandos SQL de consulta e manipulação de dados na própria ferramenta, com o uso de controle de transações;
- Realizar as alterações na base a cada inclusão/alteração/exclusão na estrutura da base.

1.2 ESTRUTURA DO TRABALHO

No segundo capítulo é abordado o conceito de banco de dados, suas funcionalidades e estruturas. É também realizada uma introdução à linguagem SQL, utilizada neste trabalho para a manipulação das bases de dados.

O terceiro capítulo apresenta o controle de versões em bases de dados, onde são demonstradas técnicas e características empregadas em uma base de dados versionável.

O quarto capítulo descreve a ferramenta proposta neste trabalho, as técnicas e ferramentas utilizadas. Neste capítulo são apresentadas também a especificação e a implementação do protótipo.

No quinto capítulo é abordada a implementação do protótipo, onde são exibidas as funcionalidades da ferramenta.

O sexto capítulo apresenta um estudo de caso, onde é criada uma situação hipotética para exemplificar uma aplicação da ferramenta.

No sétimo capítulo são descritas a conclusão e as sugestões para novos trabalhos baseados neste.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentadas as definições e características básicas de banco de dados, e realizado uma introdução à linguagem de manipulação de dados SQL.

2.1 BANCO DE DADOS

Segundo Cerícola (1995), define-se um banco de dados (BD) como sendo uma coleção de dados operacionais inter-relacionados. Estes dados são armazenados de forma independente dos programas que os utilizam, servindo assim a múltiplas aplicações de uma organização.

Analisando as partes desta definição tem-se:

a) coleção: agrupamento com repetição;
b) operacionais: vitais e estratégicos para a tomada de decisões; permanentes;
c) inter-relacionados: um banco de dados mantém um agrupamento de entidades e de relacionamentos entre estas entidades;
d) serve à múltiplas aplicações: Os dados podem ser acessados por várias aplicações. Cada uma delas define exatamente os dados que deseja manipular.

O uso de banco de dados para o armazenamento de informações traz benefícios como o armazenamento dos dados em um único local, o compartilhamento e independência dos dados e podem ser aplicadas restrições de segurança.

2.2 BANCO DE DADOS RELACIONAL

Segundo Cerícola (1995) um banco de dados relacional é uma coleção de dados organizados e integrados armazenados em forma de tabelas interligadas através de chaves primárias e estrangeiras, que constituem uma representação natural dos dados, sem imposição de restrição ou modificações, de forma a ser adequada a qualquer computador, podendo ser utilizado por todas as aplicações relevantes sem duplicidade de dados, e sem a necessidade de serem definidos em programas, pois utiliza as definições existentes nas bases de dados, através do Dicionário de Dados ativo e dinâmico.

Conforme Cerícola (1995), há três aspectos que devem ser considerados em um estudo do modelo de dados relacional:

a) aspectos estruturais: matematicamente formalizam a maneira como os dados estão organizados no modelo, sendo a formalização baseada na teoria dos conjuntos;
b) aspectos de integridade: os procedimentos para garantia de integridade de dados;
c) aspectos de manipulação: as linguagens formais e comerciais definidas para o modelo.

Conforme Machado (1996), o modelo relacional tem sua estrutura baseada em cinco conceitos: entidade, atributo, tupla, relação e chaves.

Segundo Machado (1996) define-se entidade como aquele objeto que existe no mundo real com uma identificação distinta e com um significado próprio. As entidades são representadas no modelo relacional por tabelas.

Uma entidade possui propriedades que são descritas por atributos e valores, sendo os atributos representados como colunas na entidade. Os atributos definem a forma de organização da entidade.

Uma tupla é um conjunto de pares (atributo-valor), que define uma linha da tabela, ou seja, uma ocorrência de uma entidade ou relacionamento.

Os relacionamentos são estabelecidos com a utilização de chaves primárias e chaves estrangeiras. As chaves primárias são identificadores únicos de uma tupla, sendo referenciados por outra tabela através das chaves estrangeiras.

2.3 TABELAS

Pode-se criar as tabelas no SQL Server pelas ferramentas próprias do banco, como o Enterprise Manager, ou via comandos DDL. Cada base de dados pode ter até 2 bilhões de tabelas, e cada tabela pode conter 1.024 colunas.

2.4 COLUNAS

As colunas de uma tabela são definidas de acordo com o tipo de dados que irão ser armazenados. A ferramenta Enterprise Manager não permite que após uma coluna ter sido criada na tabela, seja modificado seu tipo, para manter a integridade dos dados.

Segundo Microsoft (2001) os tipos de dados definidos no SQL Server são:

a) tipo de dados binários: suportam tipos binários e varbinários. Este tipo é utilizado para armazenar informação hexadecimal. O tipo binário armazena dados de tamanho fixo, enquanto o varbinário suporta tamanho variável de valores binários;
b) tipo de dados caracteres: possibilitam configurar a coluna para suportar letras, números e caracteres especiais. Podem ser definidos dois tipos de caracteres, caractere (*char*) e caractere de tamanho variável (*varchar*);
c) tipo de dados caracteres unicode: possibilita armazenar caracteres em uma coluna que são de múltiplas definições de caracteres;
d) tipo de dados data e hora: suporta dois tipos de dados de data, *datetime* e *smalldatetime.* Ambos os tipos suportam que os dados sejam armazenados como data e hora, somente data ou somente hora. A hora pode ser exibida em vários diferentes formatos, usando as combinações entre hora, minuto e segundo;
e) tipos de dados numéricos exatos: possibilita armazenar números com valores decimais na coluna. Podem ser definidos como exato ou aproximado. O tipo exato permite que seja definido o ponto fixo que estabelece o número de dígitos antes e depois do ponto decimal. O tipo de dados numérico aproximado permite armazenar dados numéricos com ponto flutuante, para isto deve ser definido o número de dígitos de precisão;
f) tipos de dados numéricos aproximados: este tipo inclui dois tipos de dados com ponto flutuante, *real* e *float.* O tipo *real* permite armazenar dados em uma coluna com até 24 dígitos de precisão, sendo que o *float* permite armazenamento de dados com no máximo 53 dígitos de precisão;
g) tipos de dados inteiros: podem ser definidos em quatro tipos: *bigint*, *int*, *smallint* e *tinyint*. Os tipos diferem no tamanho dos números que podem armazenar e a quantidade de espaço em disco que irão ocupar;
h) tipos de dados monetários: podem ser utilizados os tipos *money* e *smallmoney* para serem armazenados informações de moeda arredondado à décima casa decimal. Atualmente, a maioria das aplicações clientes exibem os valores monetários arredondados ao mais próximo do cento. Os tipos diferenciam-se pelos valores suportados e o tamanho em bytes que ocupam;
i) tipos de dados de texto e imagens: para armazenar um grande volume de informação em uma coluna. Possibilita serem utilizados os tipos *text* e *image*, que suportam até 2GB de dados em uma coluna;

j) tipos de dados especiais: possibilita armazenar vários tipos de dados especiais em uma coluna. Os tipos suportados são: *bit*, *cursor*, *sysname*, *timestamp* e *uniqueidentifier*;
k) tipos de dados definidos pelo usuário: possibilita ser criados tipos de dados de acordo com a definição do usuário baseado nos tipos de dados do sistema, utilizado para se ter certeza que as colunas são consistentes na base de dados.

2.5 ÍNDICES

Os índices são estruturas de dados associados às tabelas cuja principal finalidade é diminuir o tempo de acesso e recuperação de dados. Uma tabela pode ter um ou mais índices associados a ela, sendo que o índice não modifica a forma descrita do comando SQL, apenas aumenta a velocidade de recuperação e acesso aos dados (RAMALHO, 1999).

Segundo Microsoft (2001), o SQL Server utiliza dois tipos de índices, *clustered* e *nonclustered*. Índices *clustered* fazem com que os registros de tabelas sejam organizados de acordo com os campos definidos como índices *clustereds*, podendo estas ordenações ser ascendentes ou descendentes. Quando um índice é definido, devem ser identificados uma ou mais colunas na qual o índice será baseado. Esta coluna (ou colunas) é referenciada como chaves de índice. Um índice *clustered* permite configurar o SQL Server para fisicamente gravar os registros de uma tabela pela ordem de seu índice.

No SQL Server podem ser criados um total de 250 índices por tabela. Pelo fato de um índice *clustered* controlar como o SQL Server armazena os registros em uma tabela, e como elas podem ser armazenadas em uma única ordem, pode ser definido somente um índice *clustered* por tabela, e 249 índices *nonclustered* podem ser criados (ibidem).

Apesar de índices incrementar o acesso aos dados de uma tabela, criar índices demais em uma mesma tabela pode também prejudicar o acesso, devido ao fato que os índices tem que ser atualizados a cada inserção, atualização e remoção de registros. É necessário que a escolha dos campos a serem indexados, seja feita baseada em alguns pontos, como colunas com chaves primárias e estrangeiras, e colunas muito utilizadas em filtros de comandos SQL são bons candidatos a serem indexados, e colunas com muitos valores nulos e/ou valores duplicados não são bons para se indexar (ibidem).

Algumas das vantagens da utilização de índices são (MICROSOFT, 2001):

a) tipicamente proporcionam ao banco de dados a recuperação mais rápida dos dados;
b) aumentam a velocidade de comandos SQL que utilizam de várias tabelas, que fazem ordenação ou agrupamento de dados;
c) podem forçar singularidade;
d) o SQL Server mantém uma ordenação específica (ascendente ou descendente) com o índice *clustered* que está baseado nas chaves de índice.

Algumas das desvantagens da utilização de índices são (MICROSOFT, 2001):

a) devem ser seletivos (resulta em somente poucas colunas retornadas a um comando SQL) ou perdem seu valor;
b) incrementam a carga de trabalho do servidor porque os índices devem ser atualizados a cada inserção, atualização ou remoção de dados na tabela.

2.6 LINGUAGEM DE QUARTA GERAÇÃO (SQL)

Em junho de 1970, o Dr. E. F. Codd publicou um artigo intitulado *"A Relational Model of Data for Large Shared Data Banks"*. Este artigo determinou o início da mudança na filosofia de banco de dados, até então hierárquicos ou de rede. A IBM *Corporation*, *Inc*. , desenvolveu uma linguagem para usar o modelo imaginado por Codd. Esta linguagem foi batizada de SEQUEL (*Estructured English Query Language*) e posteriormente somente SQL. Hoje, a linguagem SQL é aceita como um padrão para os bancos de dados relacionais (FERNANDES, 2000).

O SQL é composto de três partes distintas:

a) linguagem de manipulação de dados (DML): possui funções como pesquisa, inserção, atualização e cancelamento de registros. Como exemplos de comandos, pode-se citar*insert* , *select*, *update* e *delete*;
b) linguagem de Definição de dados (DDL): faz a definição das tabelas, índices e visões. Pode-se citar como exemplos os comandos *create table*, *drop table*, *alter table*;
c) linguagem de Controle de dados (DCL): a segurança é o objetivo principal da DCL, com ela são controladas os privilégios dos usuários sobre tabelas ou visões. Utilizam-se comandos como *grant* e *revoke*.

3 CONTROLE DE VERSÕES

Segundo Camolesi (1996), os sistemas de gerenciamento de bases de dados têm a capacidade de adaptação, possibilitando o reconhecimento de novas situações, com a necessidade de manter um bom desempenho, um dos objetivos dos projetistas. Esta capacidade pode ser adquirida com a utilização de diferentes processos de representação, metodologias e armazenamento, proporcionando uma estabilidade relativa da base de dados, em concordância as necessidades dos projetistas.

Para Camolesi (1996), entre as metodologias mais utilizadas, encontram-se os conceitos que envolvem o controle de versões, pelo fato de várias áreas de aplicação de um software terem processos diferenciados de trabalho utilizados pelos projetistas.

A utilização de versões permite a organização estruturada de alternativas para a representação de informações em banco de dados destinadas ao armazenamento de dados complexos, indefinidos ou mutáveis. O registro histórico da evolução destas alternativas de informação não deve ter um interesse apenas de controle gerencial (produtividade), mas acima de tudo propiciar flexibilidade de trabalho aos usuários (CAMOLESI, 1996).

Neste capítulo serão descritas as motivações para a utilização de versões em bases de dados de desenvolvimento e de produção. É também dada uma atenção a uma classificação original das técnicas de evolução de esquema de dados e das estratégias de atualização nas bases de produção atingidas pela evolução.

3.1 BASES DE DADOS VERSIONÁVEIS

As bases de dados que utilizam de versões, aqui denominadas de versionáveis, tem como característica o armazenamento das informações históricas da evolução, quando o sistema realiza ações quando modificado o esquema de dados.

Conforme Camolesi (1996), é cada vez mais freqüente a utilização de versões como sendo importante para as aplicações que realizam constantes evoluções no esquema de dados e que precisam mantê-las integras. Para isso, são desenvolvidos sistemas de controle de versões, que basicamente são responsáveis pelo registro evolutivo das bases de dados. Para estruturar o controle de versões são estabelecidos modelos de versões, que abordam os aspectos conceituais e técnicos desse gerenciamento.

3.2 VERSÕES NA BASE DE DADOS DE PRODUÇÂO

Segundo Camolesi (1996), a evolução das versões em base de dados de produção são, na maioria das vezes, motivadas pelo interesse dos usuários em permitir que o sistema adquira a capacidade de obter novas funcionalidades devido as modificações das necessidades de controle informatizado dos processos da empresa. Está relacionado este interesse à complexidade do problema abordado e pela organização de trabalho dos usuários, ou seja, grandes volumes de informações são manipulados por diversos usuários com necessidades distintas.

3.3 VERSÕES NA BASE DE DESENVOLVIMENTO

As versões na base de dados de desenvolvimento e certas versões na base de dados de produção são motivadas pelo processo de evolução de esquema de dados físico/conceitual (CAMOLESI, 1996). Isto porque, o processo de evolução do esquema compreende operações críticas, que implicam em alterações das instâncias envolvidas e modificações nos programas aplicativos que utilizam estas instâncias (TRESH, 1993 apud CAMOLESI, 1996, p. 16), e para executá-las com segurança, técnicas e estratégias que utilizam de versões foram propostas por algumas por algumas pesquisas, apresentadas na próxima seção.

A relação entre instância e esquema e mesmo a associação lógica entre os elementos alterados e os inalterados de um esquema, pode gerar, inadvertidamente, alguns agravantes diretamente relacionados aos erros conceituais do modelo de dados seguido. Para este problema ser evitado, devem existir regras ou métodos no meta-esquema de dados (TRESH, 1993 apud CAMOLESI, 1996, p. 16) que mantenham o esquema de dados íntegro e as instâncias consistentes.

Para Camolesi (1996), o modelo de dados define os elementos existentes em seu esquema, um histórico de operações de alterações do esquema e as regras para a execução das ações evolutivas que geram o esquema novo. Sempre que uma alteração ocorrer envolvendo estas operações, ela deve passar por verificações de integridade nas quais são verificadas as consistências do esquema novo do modelo de dados, ou seja, um conjunto de condições, restrições e recomendações para cada uma das operações de alteração permitidas.

3.4 TÉCNICAS DA EVOLUÇÃO

Para Camolesi (1996), as técnicas de evolução do esquema não possuem denominações usuais, no entanto, a operação básica, na qual as técnicas fundamentam-se, pode ser usada como critério para classifica-las, mas nenhuma destas técnicas é detalhada a ponto de ter sua área de aplicação diminuída. Deste modo, todas são capacitadas para utilizar nas mais diferentes aplicações, mesmo porque cada técnica foi aprimorada buscando uma compreensão genérica dos problemas.

3.4.1 CRIAÇÃO/MODIFICAÇÃO

Baseada na modificação do esquema de dados em uso, para isso, é criado um esquema de dados novo utilizando uma cópia do esquema de dados já implantado. As modificações no esquema novo devem de preferência ser de pequeno volume, ou seja, pequenas e poucas alterações em relação ao esquema implantado, mesmo tendo grande influência sobre as instâncias. Normalmente, as modificações do esquema comportam operações simples (ex.: *create, delete e alter*), descritas em comandos na DDL, sobre os componentes básicos do modelo de dados seguido (CAMOLESI, 1996).

3.4.2 ADICIONAL

Esta técnica de evolução do esquema possui como característica a adição de novos elementos ao esquema de dados implantado, ou seja, a motivação desta técnica é acrescentar grandes quantidades de novos componentes na representação do esquema de dados. A adição de uma grande quantidade de componentes caracteriza sua extensão para novos contextos de representação do sistema (ibidem).

Aliada a uma abordagem própria para a implementação do projeto da base de dados, esta técnica pode oferecer meios para adição de sub-esquemas (como por exemplo esquemas suplementares (CAMOLESI, 1993) ou mesmo para a incorporação gradativa do esquema de dados em relação às instâncias que já estão armazenadas na base de dados. Para isto, requer controles especiais que permitam a adição de novos elementos ao esquema que não comprometam nenhum aspecto do processo de manipulação de instâncias.

3.4.3 ESQUEMA VERSIONADO

Propõe a criação de versões do esquema como uma forma de realizar sua evolução. Sua implementação deve permitir uma navegação entre as versões, tanto para realização de operações lógicas quanto físicas (CAMOLESI, 1996).

Dependendo do modelo de dados e da implementação, as versões antigas do esquema podem ficar inativas indefinida ou momentaneamente, ou podem ter condições iguais de uso, ou seja, a instanciação pode ser realizada em qualquer versão e, neste caso, passa e existir o conceito de esquema corrente (formado pelas versões de esquema requisitadas) e com as instâncias sendo reconhecidas somente quando sua definição está no esquema em uso corrente (ibidem).

As versões podem ser criadas sobre o esquema completo ou sobre uma parte, dependendo do grau de evolução que sofrerá o esquema corrente, do tipo do modelo de versões seguido e da implementação do controle de criação de versões (ibidem).

Um controle rigoroso deve ser implementado para o gerenciamento do esquema, que permita a criação de versões em coerência com as operações de alteração do esquema em relação ao modelo de dados. Sendo, talvez, um processo de duração longa e de grande complexidade, podendo-se optar pela criação simultânea de alternativas para uma versão, ou seja, o desenvolvimento paralelo de possíveis versões do esquema (ibidem).

No final do processo de criação, as alternativas são confrontadas para ser escolhida aquela que melhor representa o empreendimento em sua nova definição e que se tornará uma versão do esquema. As alternativas podem passar também por operações onde busca-se aglutinar as melhores inovações ou soluções apresentadas em cada alternativa para a formação final de uma versão do esquema (ibidem).

3.5 ESTRATÉGIAS DE PROPAGAÇÃO NAS INSTÂNCIAS

Segundo Bjornerstedt (1989), pode-se encontrar classificações aparentemente estabelecidas para as estratégias de propagação, que descrevem a abordagem usada para a utilização do esquema de dados novo, pode-se encontrar também muitas variações destas estratégias, inclusive das que utilizam versões, em adaptações para problemas específicos. Sendo que estas estratégias enquadram-se em categorias descritas a seguir.

3.5.1 COPIANDO

A cópia apresenta uma abordagem simples porem muito utilizada. Nela, após o estabelecimento do esquema novo, realiza-se a transposição imediata das instâncias envolvidas na evolução, ou então, realiza-se a transposição incremental de partes das instâncias envolvidas até que todas tenham sido alteradas. A transposição é realizada através da criação de cópias de seus dados para a nova situação (do esquema de dados em uso para o esquema de dados novo) (CAMOLESI, 1996).

A desvantagem nesta abordagem é o tempo consumido para a finalização das cópias de acordo com o novo esquema de dados, o que pode depender da implementação do sistema de gerenciamento (ibidem).

3.5.2 ATUALIZANDO

Nesta abordagem, a passagem para um esquema novo ocorre com a conversão de suas instâncias envolvidas. Quando a conversão das instâncias é realizada imediatamente após a criação do esquema novo, recebe a denominação de conversão imediata. A desvantagem da conversão imediata está no tempo consumido para a finalização de todas as conversões em uma única vez, de acordo com o esquema novo.

Outra opção é de converter as instâncias durante sua manipulação, através da conversão incremental. A cada informação acessada, é reconhecido seu esquema de dados e verificado a existência de alguma indicação de evolução. Caso seja encontrada, o sistema realiza a adaptação lógica da informação de acordo com o seu esquema novo, que age como um filtro para criar a perspectiva de instâncias modificadas (CAMOLESI, 1996).

Esta estratégia compromete a velocidade de execução de todas as operações de manipulação de dados, uma vez que as informações demonstradas aos usuários devem ser convertidas de acordo com o esquema novo (ibidem).

3.5.3 VERSIONANDO

Nesta estratégia, um novo esquema de dados, ao ser instanciado, leva à criação de versões das instâncias atingidas. Dependendo da implementação, as versões das instâncias

podem ter diferentes níveis de flexibilidade para a manipulação ou podem ter condições iguais de uso (CAMOLESI, 1996).

As versões de instâncias podem ser criadas gradativamente pelo usuário, desde que monitoradas pelo sistema, ou podendo ser geradas automaticamente pela cópia/conversão, imediata ou incremental, de uma versão de instância para outra, desde que utilizado o esquema de dados correspondente (ibidem).

Esta técnica não apresenta desvantagens significativas, apenas requer um nível maior de controle, normalmente realizado por um sistema de gerenciamento de versões que normalize as atividades de criação, manipulação, remoção e consulta de versões de instâncias que podem habilitar a base de dados (ibidem).

4 DESENVOLVIMENTO DO TRABALHO

Neste capítulo é descrito o sistema a desenvolvido neste trabalho, as técnicas e ferramentas utilizadas e a análise essencial do sistema.

4.1 SISTEMA PROPOSTO

Para que se tenha uma base de dados versionável, é necessário armazenar todas as modificações realizadas no decorrer da evolução do seu esquema. Para isso, o primeiro passo a ser dado, é no momento que o usuário fizer o *login* na base de dados, a ferramenta verifique se as tabelas que contém a estrutura da base estão criadas, e caso não estejam, efetua a criação.

As tabelas de estrutura da base consistem em tabelas que armazenam as entidades, seus atributos, as bases de produção (clientes), as alterações exportadas aos clientes, as versões da base de dados e as alterações realizadas na estrutura da base.

O projetista deve cadastrar a versão corrente da base e clientes. A cada cadastro de nova versão, deve ser fechada a versão anterior, obtendo-se assim o controle das modificações por versões da base.

Com a versão corrente informada, o projetista está habilitado a definir as entidades e seus atributos. Isto é feito de maneira visual na ferramenta, onde as informações disponibilizadas são convertidas em linguagem SQL e aplicadas fisicamente na base de dados. Podem ser criados, alteradas e removidas tabelas, campos e índices. A cada modificação realizada, é armazenada a alteração na tabela de *logs* da ferramenta.

Opcionalmente o projetista pode informar se uma tabela, campo ou índice, é específico de um cliente, neste caso, estas alterações não serão aplicadas nas bases de clientes no qual não pertencem.

Com o histórico da evolução da base, podem ser gerados *scripts* de exportação das modificações da estrutura, informando o intervalo de alterações desejadas. Para gerar este *script*, pode-se informar se ele é específico de algum cliente, ou pertencente à todos os clientes cadastrados. Aplicam-se verificações nas alterações informadas que se deseja exportar, para verificar se caso um cliente tenha sido informado, sua base já não foi atualizada

com alterações dentro do conjunto estabelecido, e caso as alterações a serem exportadas são a todos os clientes, se existe alterações específicas de clientes, ou se clientes possuem bases com alterações já atualizadas pertencentes às informadas. Havendo restrições, o sistema informa o projetista da situação, e orienta como proceder para resolve-la.

Quando é gerado um *script* de exportação, a tabela que contém as informações dos clientes e suas alterações exportadas, é atualizada para obter o controle da situação atual da base de cada cliente específico.

Com este *script* gerado, tanto o projetista quanto o cliente (responsável pela TI da empresa), podem atualizar a base de produção, executando a ferramenta e no seu local de atualização, importar o *script* e confirmar, a ferramenta então irá aplicar todas as modificações na ordem implementada na base de desenvolvimento, gradativamente até as modificações estarem todas aplicadas.

4.2 TÉCNICAS E FERRAMENTAS UTILIZADAS

Para a implementação do protótipo foi utilizado o ambiente de desenvolvimento Delphi 7.0, sendo a especificação realizada utilizando a análise essencial, com a modelagem da estrutura do aplicativo e outros gráficos através da ferramenta CASE PowerDesigner. O banco de dados utilizado para o desenvolvimento é o SQL Server 2000.

O Delphi foi escolhido para o desenvolvimento da ferramenta tendo em vista a experiência anterior do autor na sua utilização. Além disso a ferramenta possui componentes de interação com banco de dados que permitem a expansão da mesma para obtenção de acesso simplificado aos principais bancos de dados do mercado.

4.2.1 AMBIENTE DE DESENVOLVIMENTO DELPHI

É um ambiente de desenvolvimento de aplicações, orientado a objetos, que possibilita o desenvolvimento de aplicações diminuindo o trabalho de codificação.

Segundo Swan (1996), Delphi é um sistema de desenvolvimento rápido de aplicativos, ou RAD, é um inteligente gerador de códigos, um projetista de aplicativos visuais e uma ferramenta de banco de dados com uma interface extraordinária, simples de aprender e poderosa de usar.

4.2.2 ANÁLISE ESSENCIAL DE SISTEMAS

Conforme Machado (1996), a análise essencial relaciona-se diretamente com eventos, que causam a reação do sistema, tendo o sistema um conjunto de reações que respondem aos eventos. É formado por Lista de Eventos, Diagrama de Contexto, Diagrama de Fluxo de Dados (DFD) e Modelo Entidade Relacionamento (MER conceitual e físico).

A lista de eventos compõe o primeiro passo na especificação de um sistema, que auxilia a delimitar os problemas de que estamos tratando. A lista de eventos é uma lista textual dos estímulos do ambiente externo no qual o sistema deve responder (ibidem).

O diagrama de contexto é elaborado baseado na lista de eventos, onde é representado o sistema por um único processo e suas interações com as entidades externas (ibidem).

O DFD é o diagrama de contexto separado pelos eventos, representa os processos e o fluxo de dados entre eles (ibidem).

MER demonstra uma visão simples da estrutura das entidades do sistema, abstraindo os detalhes funcionais (ibidem).

Dicionário de dados provém do MER formando uma listagem de informações sobre os componentes do sistema. Os dicionários de dados dispõem a informação em forma de texto com a finalidade de auxiliar a informação mostrada no DFD (ibidem).

4.2.3 POWERDESIGNER 10.0

É uma ferramenta CASE para modelagem de sistema, permite desenvolver uma estrutura lógica de um banco de dados que é independente do software de desenvolvimento ou da estrutura de armazenamento dos dados. Foi utilizado o PowerDesigner 10.0 para desenvolver o diagrama de contexto, diagrama de fluxo de dados e o MER.

4.2.4 SQL SERVER 2000

Segundo Microsoft (2001), o SQL Server incluí muitas características que o fazem um poderoso sistema de gerenciamento de base de dados para as redes empresariais e pequenas redes. Estas características incluem tudo desde suportea uma grande variedade de sistemas operacionais para integração com Windows 2000 e aplicações servidoras da Microsoft.

Algumas características são descritas a seguir:

a) suporte a múltiplas plataformas: Windows 98, ME, NT 4.0, 2000 Professional ,2000 server e XP;
b) integração com Microsoft .NET Enterprise Servers;
c) escalabilidade: o sistema de gerência do banco de dados pode crescer de acordo com a companhia;
d) replicação: a possibilidade de ter mais de um SQL Server na mesma companhia;
e) gerenciamento centralizado: pode-se gerenciar todos os servidores utilizando a ferramenta SQL Server Enterprise Manager;
f) tarefas automáticas: a habilidade de agendar trabalhos, como por exemplo agendar backup de base toda quinta-feira às 20:00 horas.

4.3 ESPECIFICAÇÃO

Nesta seção são descritas as atividades que compõem a fase de especificação do projeto, utilizando a análise essencial, descrita na seção 4.2.2 deste trabalho.

4.3.1 LISTA DE EVENTOS

A especificação de um sistema deve ser iniciada pela descrição dos eventos que ocorrem nele. A seguir estão numerados os eventos.

1. Projetista faz *login* na base de dados.
2. Projetista cadastra versão da base.
3. Projetista cadastra clientes (bases de produção).
4. Projetista define entidades.
5. Projetista fecha versão.
6. Projetista gera *script* de exportação das modificações da base.
7. Projetista ou cliente atualiza base com o *script* gerado.

4.3.2 DIAGRAMA DE CONTEXTO

Com base na lista de eventos relacionada anteriormente, foi desenvolvido o diagrama de contexto (Figura 1) com o uso da ferramenta CASE PowerDesigner.

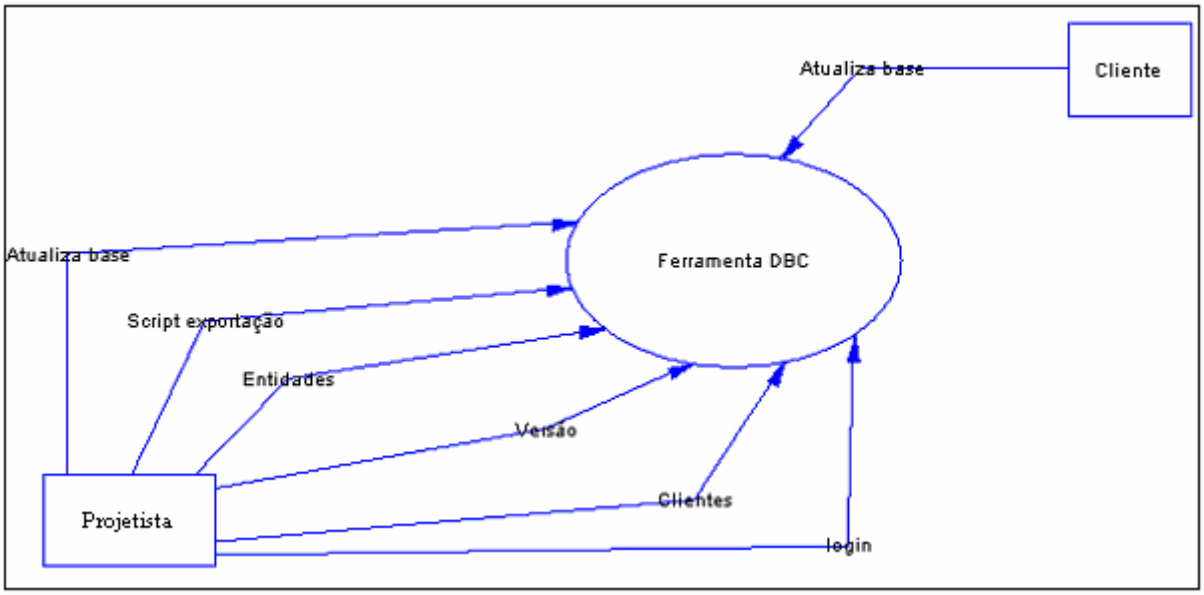

Figura 1 - Diagrama de contexto

4.3.3 DIAGRAMA DE FLUXO DE DADOS

A base de dados que se deseja manipular, e a identificação do usuário, são informados no *login* da ferramenta, demonstrado na figura 2.

Figura 2 - DFD do evento número 1

A figura 3 representa o cadastramento de versões da base onde o projetista define a versão corrente do sistema e fecha as versões anteriores.

Figura 3 - DFD do evento número 2

O evento exibido na figura 4, representa o cadastramento de clientes da *software house*, estes podendo ser utilizados na definição do esquema de dados e controle de atualizações das bases de produção.

Figura 4 - DFD do evento número 3

A criação, alteração e exclusão de entidades em uma base de dados, e as tabelas utilizadas para armazenar a estrutura do esquema de dados, são exibidos no DFD da figura 5.

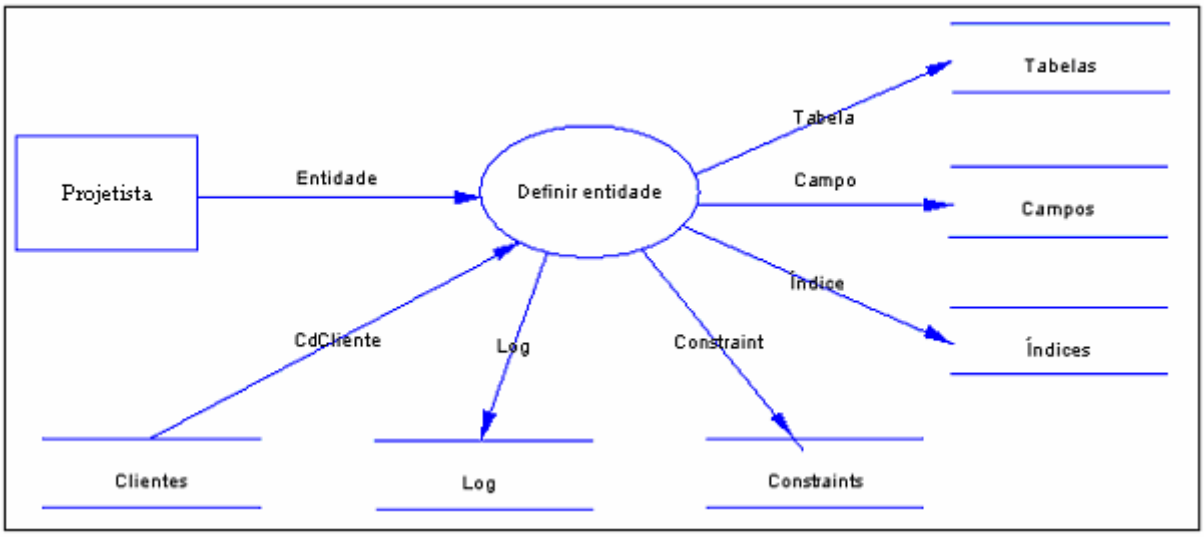

Figura 5 - DFD do evento número 4

O controle de estados das versões da base de dados é representado no DFD figura 6.

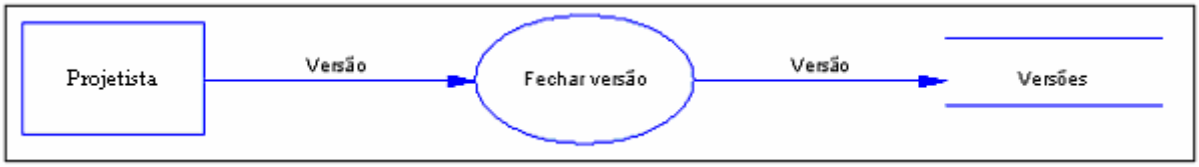

Figura 6 - DFD do evento número 5

A criação de *scripts* de exportação, gerados de acordo com o(s) cliente(s) a ser atualizado(s), e o intervalo de modificações realizado no esquema de dados a ser exportado, é demonstrada no DFD figura 7.

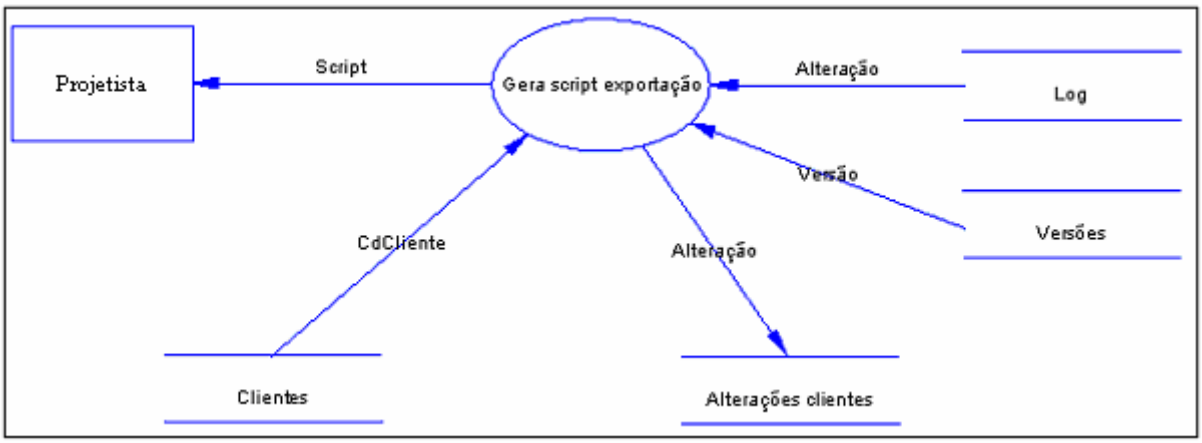

Figura 7 - DFD do evento número 6

No DFD da figura 8 é demonstrada a atualização de bases de dados de produção, onde o conjunto de alterações definidas no *script* de exportação é aplicado, gerando um novo esquema de dados.

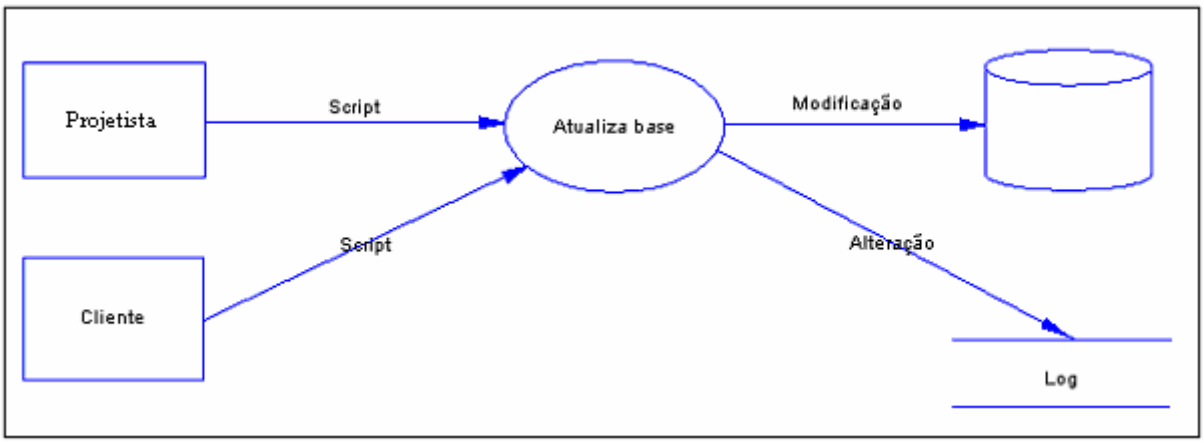

Figura 8 - DFD do evento número 7

4.3.4 MODELO ENTIDADE RELACIONAMENTO

Esta etapa exibe um MER do modelo conceitual (Figura 9) e do modelo físico (Figura 10), que representam as informações do sistema.

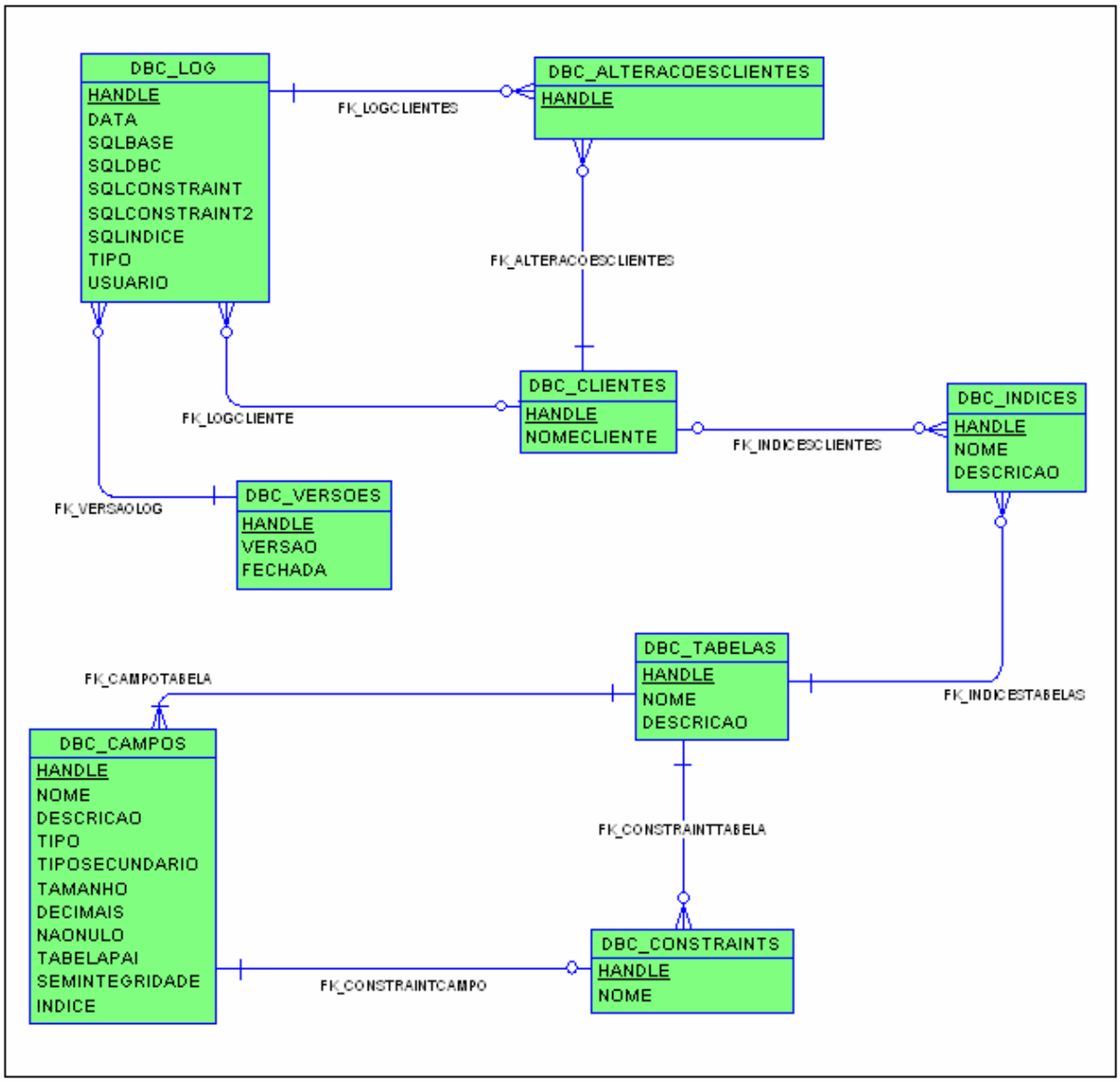

Figura 9 - MER - Modelo conceitual

Os clientes cadastrados no sistema, que podem ter definições de entidades e/ou atributos específicos, e também utilizados para a geração de *scripts* de exportação, são armazenados na tabela DBC_CLIENTES. Para o controle da versão das bases de dados de produção, é utilizada a tabela DBC_ALTERACOESCLIENTES, que contém todas as alterações já implantadas em cada cliente. Os registros desta tabela são atualizados quando são gerados scripts de exportação aos clientes. Cada alteração registrada nesta tabela faz referência a uma alteração realizada na base por um projetista.

A tabela DBC_TABELAS armazena a lista de tabelas existentes na base de dados, sendo a estrutura de seus campos armazenados na tabela DBC_CAMPOS e seus índices na tabela DBC_INDICES. As chaves estrangeiras que garantem a integridade de relacionamentos entre as tabelas são referenciadas na tabela DBC_CONTRAINTS.

Na tabela DBC_LOG fica registrado o histórico da evolução do esquema de dados, com os comandos SQL necessários para a propagação das alterações em bases de produção, sendo atualizada a cada modificação implementada um uma base por um projetista. Para cada alteração é informada em qual versão da base de dados ela foi realizada, sendo as versões referenciadas em DBC_VERSOES.

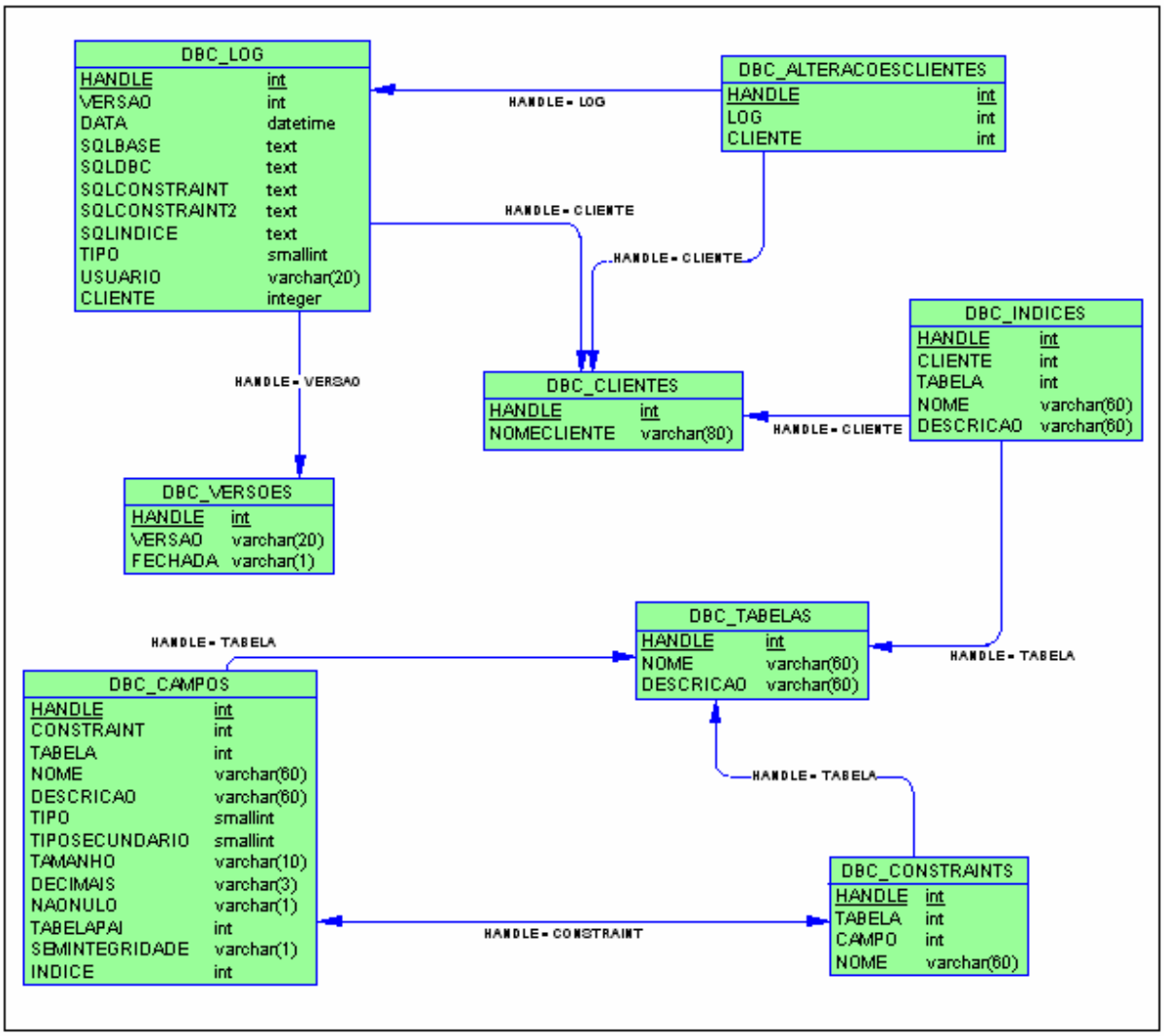

Figura 10 - MER - Modelo físico

5 DESCRIÇÃO DA IMPLEMENTAÇÃO

Neste capítulo são descritas as funcionalidades do sistema. Inicialmente é apresentado como é feito o acesso ao banco de dados, através do DBExpress. Posteriormente são apresentadas as funcionalidades da ferramenta desenvolvida.

5.1 UTILIZANDO O DBEXPRESS

O DBExpress é uma categoria de componentes do Delphi, utilizado para acesso a banco de dados, disponibilizado na versão 6.0 e posteriores do Delphi. O DBExpress proporciona uma série de vantagens com o seu uso, como por exemplo o armazenamento das alterações realizadas no banco em *buffer* (aplicadas fisicamente no banco somente quando explícito pelo projetista) e abstração de configurações para acesso ao banco à nível de usuário do sistema que o utiliza.

O DBExpress possibilita estabelecer conexão com os mais utilizados banco de dados do mercado. Na propriedade DriverName do componente SQLConnection, seleciona-se o *driver* nativo do banco de dado desejado, com as opções DB2, Informix, Interbase, MSSQL, MySQL e ORACLE.

Na propriedade de parâmetros (Figura 11) com o *driver* MSSQL selecionado, informa-se o HostName (máquina onde o banco está instalado), DataBase (a base que irá ser administrada), o usuário e a senha que irão fazer a conexão. Outros parâmetros devem ser informados como o tamanho de um campo *blob* na base, o modo de autenticação e o nível de isolamento de uma transação no banco de dados.

Com estas informações e definindo o SQLConnection como ativado, a conexão com o banco é estabelecida. Uma das desvantagens do uso do DBExpress, é que o acesso das *querys* à base de dados é unidirecional, ou seja, a navegação dos registros é somente em um sentido. Para resolver este problema é necessário utilizar quatro componentes: SQLQuery, DataSetProvider, ClientDataSet e DataSource.

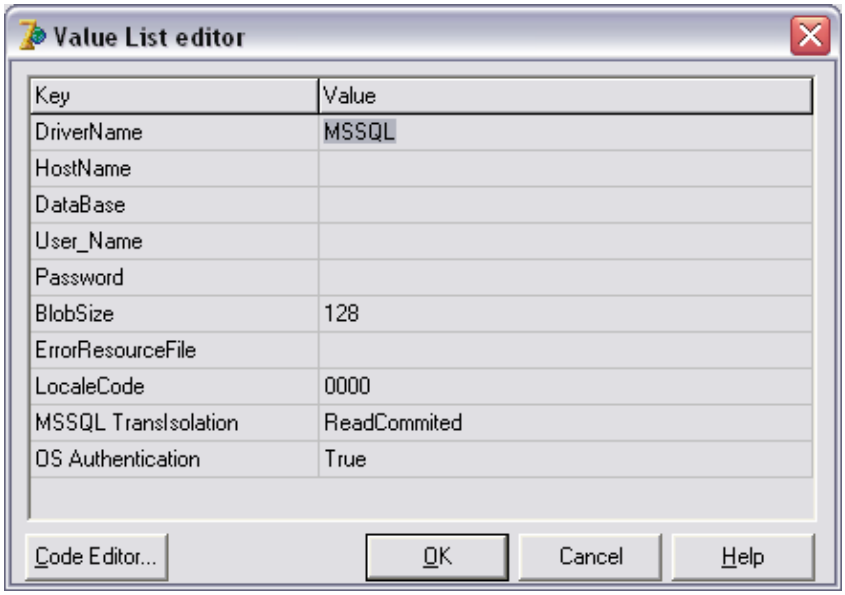

Figura 11 - Parâmetros do SQLConnection

No SQLQuery, são informados os comandos DML, sendo que à este componente está referenciado o DataSetProvider, que irá armazenar em *buffer* as alterações de dados aplicados à tabela relacionada no SQLQuery. Com o ClientDataSet são aplicadas as alterações nos dados da tabela em questão, quando por exemplo em uma confirmação de inserção de um registro na tabela, ou uma seqüência de alterações em fila no *buffer*, estes dados não estão ainda inseridos fisicamente na base de dados (armazenada em buffer), é necessário então aplicar os *updates* com o método *ApplyUpdates* do componente ClienteDataSet. Caso tenha algum erro de integridade dos dados neste momento, pode ser configurado para que o componente não permita aplicar as modificações levantando uma mensagem de erro. E com os componentes ClienteDataSet refenciado em DataSetProvider, pode-se então ter controle bidirecional sobre os dados da tabela.

5.2 CONECTANDO NA BASE DE DADOS

Para se conectar a um banco de dados deve-se informar o servidor (*host*) e o nome da base desejada, ilustrado na figura 12. A base deve estar criada no banco de dados, com suas configurações já estabelecidas (ver apêndice II - Criação de bases no SQL Server 2000). Deve ter também sido informados na base em questão os usuários com acesso a seu esquema de dados. O usuário e sua senha são informados (Figura 13) após informados o servidor e a base.

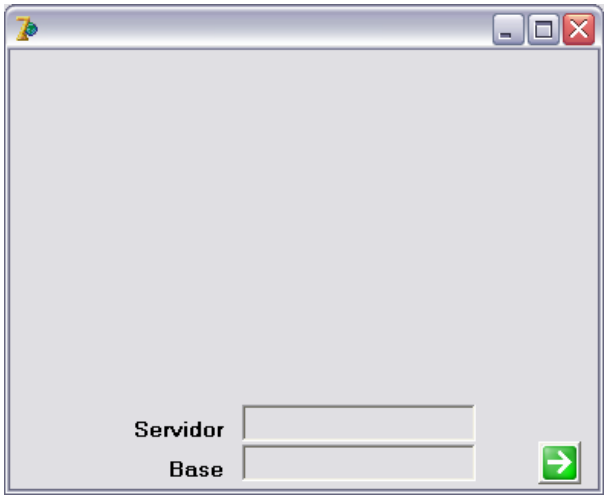

Figura 12 - Parâmetros do banco de dados

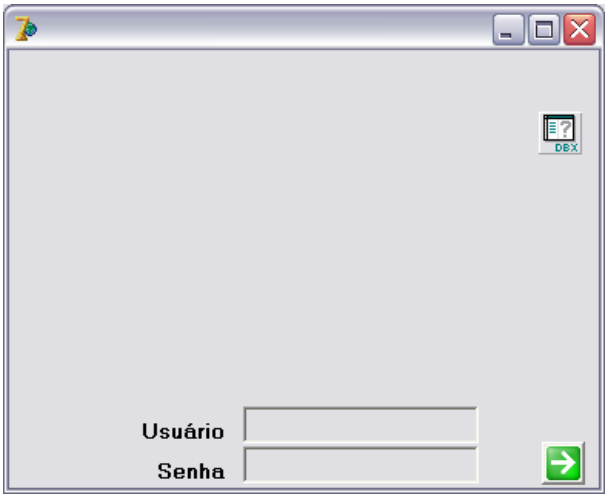

Figura 13 - Parâmetros da base de dados

Após estes parâmetros terem sido definidos, a ferramenta verifica a validade dos dados informados, garantindo o acesso do usuário à base.

É feita então uma verificação na base conectada, para analisar se as tabelas que contém a estrutura do esquema de dados estão criadas (ver capítulo 4.3.4). Caso elas não tenham sido criadas, a ferramenta cria estas entidades fisicamente na base de dados selecionada.

Ao entrar no formulário principal da ferramenta (Figura 14), é verificado se a versão corrente da base está informada, caso não esteja, ou se o projetista deseja cadastrar uma nova versão corrente, isto é realizado na página “Versões” do formulário principal. Para ser cadastrada uma nova versão, e caso tenha versão uma versão corrente, e necessário “fechá-la”, o que significa que em uma versão fechada não há mais alterações na estrutura da base a

serem implementadas. Os clientes (bases de produção) devem ser cadastrados na página "Clientes" deste mesmo formulário, onde são informados o nome e a versão corrente da estrutura em que se encontra.

5.3 DEFININDO AS ENTIDADES

No formulário principal, podem ser criadas, modificadas e removidas as tabelas. A figura 14 ilustra as tabelas da base, onde na sua criação devem ser informados o seu nome, sua legenda (o nome que a tabela terá visualizado na árvore), e se ela é específica de um cliente.

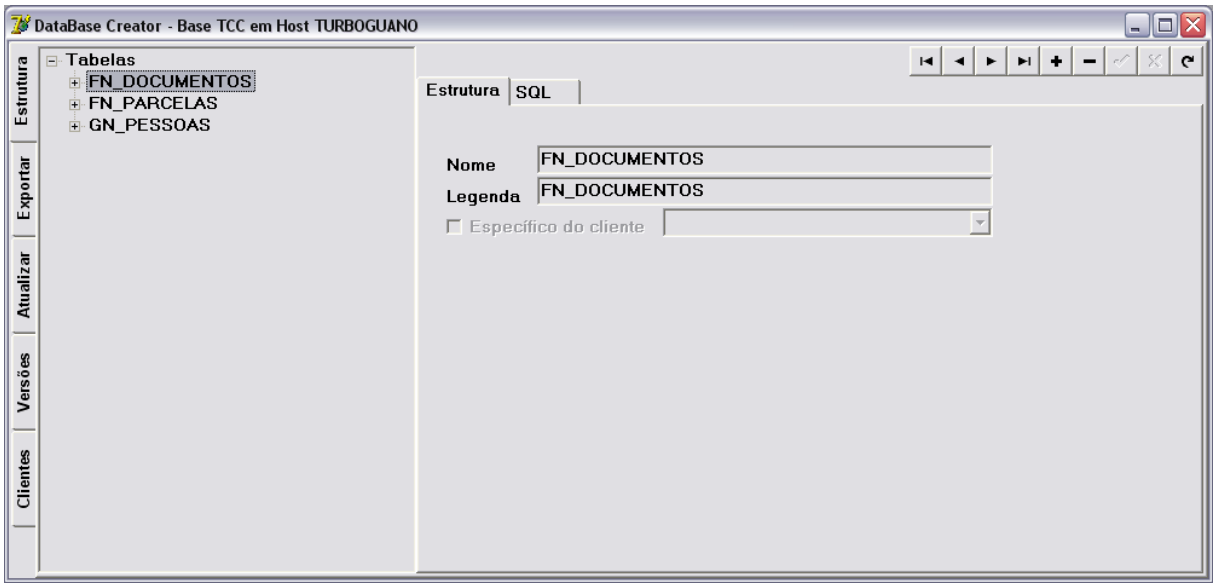

Figura 14 - Cadastro das tabelas

Toda a estrutura das tabelas, seus campos e índices é disposta em forma de árvore hierárquica (ver figura 15), sendo exibidos os campos e índices abaixo da tabela no qual pertencem.

Uma modificação no esquema de dados é realizada visualmente na ferramenta e convertida em comandos SQL, quando confirmada se não contiver erros de integridade é aplicada fisicamente na base de dados. Para cada modificação realizada, seja ela de tabelas, campos ou índices, a ferramenta grava um *log* detalhado com os comandos SQL utilizados, qual usuário fez a modificação e a data da mesma. Estas informações são necessárias para que a evolução das modificações seja implantada fisicamente em uma base de dados de produção.

Para este trabalho foram escolhidas características de ambas as técnicas de evolução modificação e adicional, descritas na seções 3.4.1 e 3.4.2. Sendo a técnica de modificação utilizada quando uma instância já existente no esquema de dados é alterada, e a técnica adicional quando da criação de novas instâncias

Por definição toda tabela criada tem um campo chamado *handle*, que por definição é a chave primária das tabelas criadas na ferramenta. Tem-se também a opção de criar um campo tipo inteiro na tabela, e na página "Inteiro" das definições de campos do formulário principal, definir este campo como sendo a chave primária da tabela.

Nas definições das tabelas e seus campos, há a possibilidade de informar se tabelas ou campos são específicos de um cliente, ou seja, modificações que serão somente propagadas na base de produção do cliente informado. Isto é definido no campo "específico do cliente" na estrutura da tabela ou campos.

Selecionando uma tabela, e abrindo sua árvore hierárquica, podem ser definidas as colunas (campos) pertencentes a sua estrutura. Para a definição de campos, há uma página para cada tipo de campo, onde devem ser informadas as características próprias necessárias para que ele seja criado. Na figura 15, um exemplo de criação de um campo que faz referência com outra tabela. Informa-se o nome do campo e a sua legenda, se ele é específico de um cliente e se permite valores vazios em seu conteúdo. Informa-se então a tabela na qual ele faz referência e se possui integridade referencial com esta tabela. Se o campo "sem integridade" (exibido na figura 15) estiver checado, não será criada a *constraint* para forçar a integridade entre o relacionamento.

Neste caso de campo do tipo relacionamento, quando é confirmada a sua inserção, a ferramenta cria o campo do mesmo tipo da chave primária da tabela no qual ele faz referência, e caso esteja informado para ter integridade referencial, é criada automaticamente o relacionamento físico entre este campo e a tabela referente (*constraint*).

Figura 15 - Cadastro de campos

Para a elaboração deste trabalho foram implementados os tipos de dados "relacionamento", inteiro, número, moeda, string, data e binário, não foram aqui implementados outros tipos de dados como *blobs* (texto ou imagem) e tipos especiais definidos pelo usuário. Para cada um dos tipos implementados, em sua página, devem ser informados os parâmetros necessários para que eles sejam criados de acordo com as definições do banco de dados SQL Server.

Um campo também pode ser modificado. Por exemplo, podem ser alteradas as definições de tamanho e tipo, desde que estas não afetem a integridade da base. Neste caso a ferramenta faz uma verificação de restrições para se manter a integridade, não permitindo, por exemplo, que um campo do tipo string seja alterado para o tipo inteiro.

Tendo como exemplo a alteração de tipo de um campo de inteiro para string, a ferramenta cria uma tabela temporária na base de dados onde são copiados os dados pertencentes à tabela sendo modificada, da coluna em questão. A coluna então é excluído da tabela e criada novamente com seu novo tipo, e os dados então transferidos de volta para a tabela modificada.

Na árvore de índices, são exibidos os campos com índices criados. No formulário de índices são listados os campos da tabela selecionada habilitados a serem criados índices (pode ser selecionado um ou mais campos para ser criado um índice composto). Para remover um

índice, deve-se selecionar o campo desejado que o índice ligado a ele será removido do sistema. A manutenção de índices é ilustrada na figura 16.

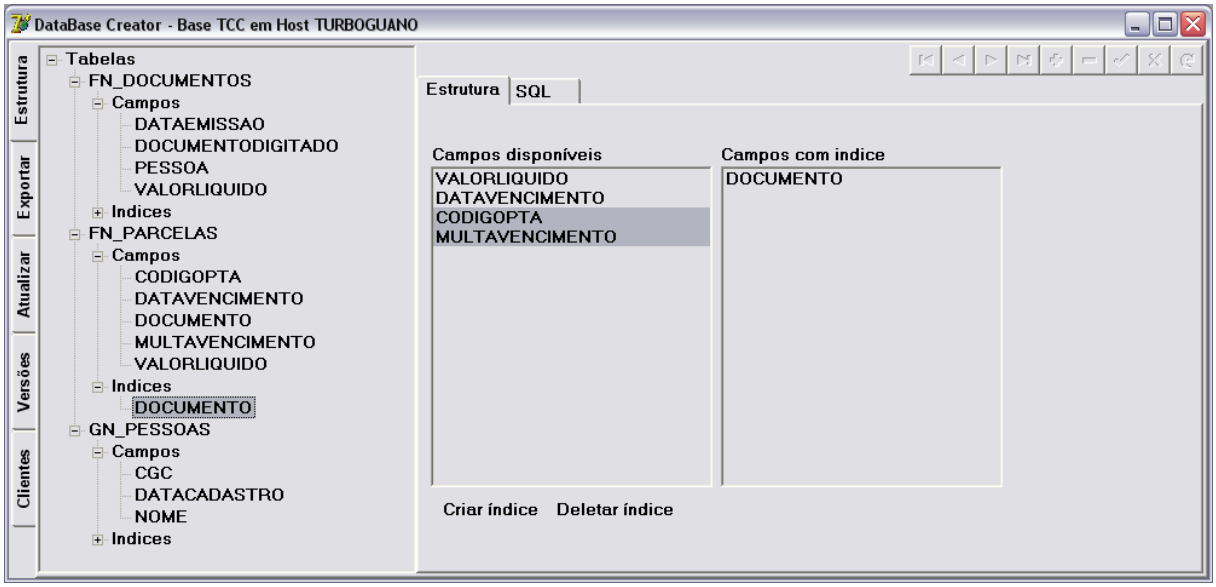

Figura 16 - Cadastro de índices

5.4 EXECUTANDO COMANDOS SQL

Comandos SQL podem ser executados na página “SQL” do formulário principal (Figura 17). Pode-se iniciar uma transação pelo botão “Start Transaction” antes de um ou mais comandos SQL serem executados. Quando desejado, é possível confirmar as operações realizadas na transação pelo botão “commit” ou cancelar através do botão “rollback”, fazendo assim com que as entidades retornem ao estado anterior à transação ter sido iniciada. A ferramenta permite a execução de comandos DML e DDL.

O projetista tem a possibilidade também de definir a opção “Gerar log”, que faz com que seus comandos SQL sejam gravados na *log* da ferramenta, possibilitando assim que estes comandos sejam executados também nas bases de dados de produção.

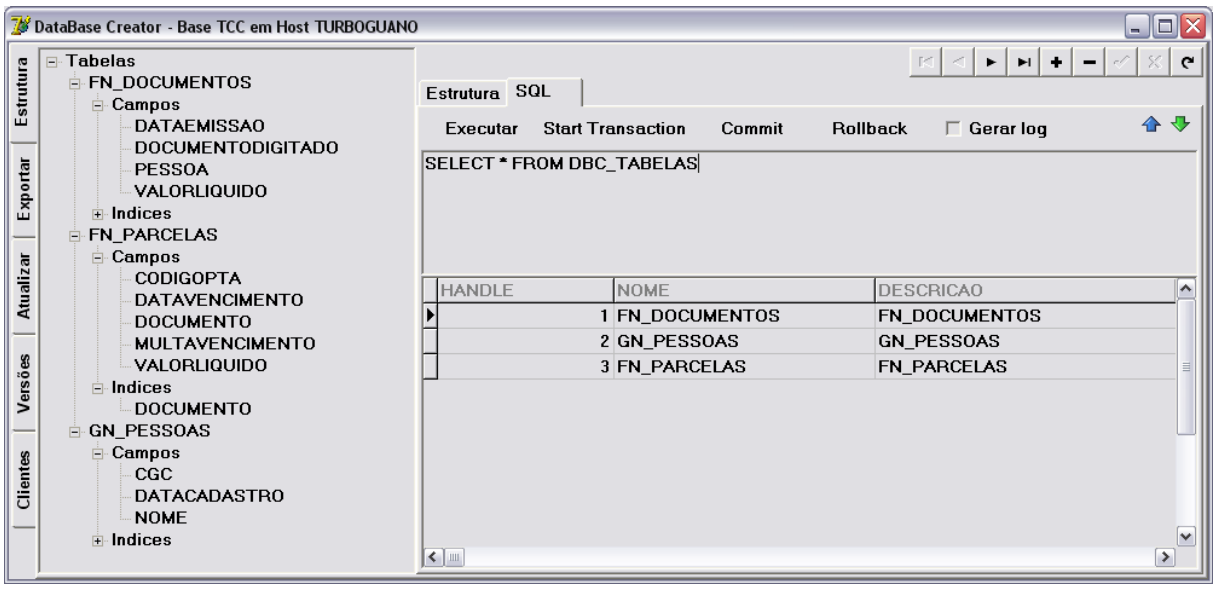

Figura 17 - Comandos SQL na ferramenta

5.5 GERANDO SCRIPTS DE EXPORTAÇÃO

Para gerar um *script* de exportação das modificações do esquema de dados, implementado na página “Exportar” do formulário principal (Figura 18), deve-se informar o local e o nome do arquivo e se ele será gerado para um cliente em específico ou para todos os clientes cadastrados. Deve ser informado também se deseja gerar um *script* das modificações de uma versão cadastrada na ferramenta, e ainda caso não tenha sido informada a versão, pode-se filtrar as alterações desejadas a serem exportadas informando a alteração inicial (campo “Handle Inicial”) em conjunto com a definição de “Select Especial”, por exemplo, deseja-se exportar as alterações de número 10 até 14, deve ser informado o handle inicial 10 e o select especial como “HANDLE <= 14” ou não informar o handle inicial e o select especial como “HANDLE >= 10 AND HANDLE <= 14”.

O gráfico superior do formulário apresentado na figura 18, informa todas as alterações realizadas na base. A cor azul representa as alterações de uma versão em aberto, a cor verde representa as alterações específicas de clientes e caso esteja em vermelho significa alterações de uma versão fechada.

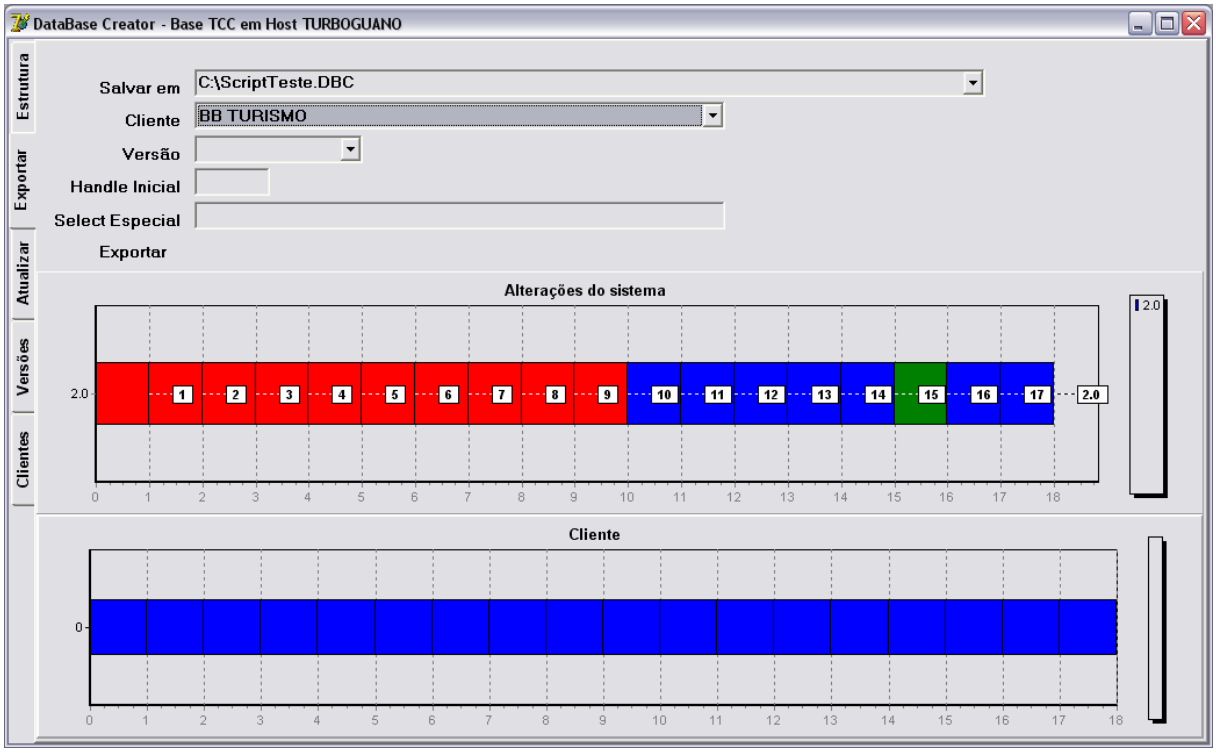

Figura 18 - Script de exportação

No gráfico inferior é exibido quando um cliente específico está informado. Nele são demonstradas em vermelho as alterações já exportadas a este cliente, e em azul as alterações ainda não existentes na base do cliente.

A ferramenta verifica se há restrições nas alterações que se deseja exportar para as bases de produção a serem atualizadas. A seguir são descritas as restrições de exportação e como a ferramenta procede para tratá-las:

a) com cliente específico informado, no conjunto de alterações especificadas contém alterações já atualizadas no cliente, neste caso, a ferramenta retina as alterações já exportadas ao cliente do *script*;
b) sem cliente específico informado, no conjunto de alterações especificadas, a ferramenta verifica se há alterações exclusivas de clientes, e caso sim, informa ao projetista para que aos clientes que tenham alterações específicas sejam gerados *script* separadamente;
c) sem cliente específico informado, no conjunto de alterações especificadas, a ferramenta verifica se tem clientes com as alterações já exportadas, e caso sim, a ferramenta informa ao projetista que para estes clientes sejam gerados *scripts* separadamente.

Quando um *script* é gerado, para os clientes sem restrições, é atualizada a tabela que contém as informações das modificações exportadas a cada cliente. É obtido assim o controle da situação de cada base de produção, possibilitando gerar corretamente *scripts* para a atualização de cada uma.

O algoritmo utilizado para gerar os scripts de exportação é exibido no quadro 1.

```
// ---- Busca as alterações informadas ---- //

Se estiver informada a versão
      Busca as alterações do sistema da versão informada
      Se tiver cliente informado, busca alterações especificas
Fim se

Se versão não informada
      Verifica se foi informado a alteração inicial a ser exportada
      Verifica se foi informado um grupo específico de alterações
Fim se

// ---- Verifica as restrições do script a ser gerado ----- //

Se não tiver cliente informado
      Verifica se tem alterações específicas de cliente
            Retira o(s) cliente(s) com alterações específica dos clientes a atualizar exportações
      Fim se

      Se tiver alterações específicas para clientes
            Informa ao usuário para gerar scripts específicos para este(s) cliente(s)
            Retira o(s) cliente(s) com alterações especficas dos clientes a atualizar exportações
            Retira estas alterações do script a ser gerado
      Fim se


      Verifica se para algum cliente as alterações já foram exportadas
            Retira o(s) cliente(s) com alterações já exportadas dos clientes a atualizar exportações
            Informa ao usuário para gerar scripts específicos para este(s) cliente(s)
      Fim se
Fim se


Se tiver cliente informado
      Verifica se o cliente já tem alteração exportada
      Retira alterações já exportadas ao cliente
Fim se

Gera o script de alterações

Atualiza a tabela que contém as exportações efetuadas aos clientes
```

Quadro 1- Algoritmo para gerar o script de exportação

5.6 ATUALIZANDO UMA BASE DE PRODUÇÃO

Para atualizar uma base de produção, é recomendado que seja feito um *backup* da base antes da operação de atualização. Tanto o projetista da base de desenvolvimento acessando remotamente o cliente quanto o responsável pelo setor de TI do cliente podem realizar a atualização. Deve-se entrar na ferramenta desenvolvida neste trabalho, e na página "Atualizar" do formulário principal somente seleciona-se o arquivo de *script* e confirmando a operação.

A ferramenta irá criar, modificar ou remover as entidades na mesma ordem em que foram realizadas na base de desenvolvimento, e de acordo com o conjunto de alterações informadas no *script* gerado, apresentado na tela de *log* todas as alterações implantadas na base de dados com a atualização aplicada.

A ferramenta desenvolvida neste trabalho, não permite uma navegação de versões de uma base de dados, ou seja, uma base após ser atualizada, não é possível voltar para a versão anterior à atualização. Isto pode ser feito somente restaurando um *backup* da base de dados.

6 ESTUDO DE CASO

Neste capítulo é criada uma situação fictícia simplificada para demonstrar as funcionalidades da ferramenta proposta neste trabalho. É demonstrada a evolução dos esquemas de dados das bases envolvidas neste estudo de caso, para exemplificar o gerenciamento de versões em bases de desenvolvimento e de produção.

Uma *software house* com a intenção de expandir suas áreas de atuação, desenvolveu um sistema para controle de vacinas de clínicas veterinárias. A primeira versão deste sistema sendo comercializada é denominada de versão 1.0. Tem como funcionalidades o cadastramento de vacinas, dos clientes e seus animais, e o gerenciamento das vacinas aplicadas em cada animal.

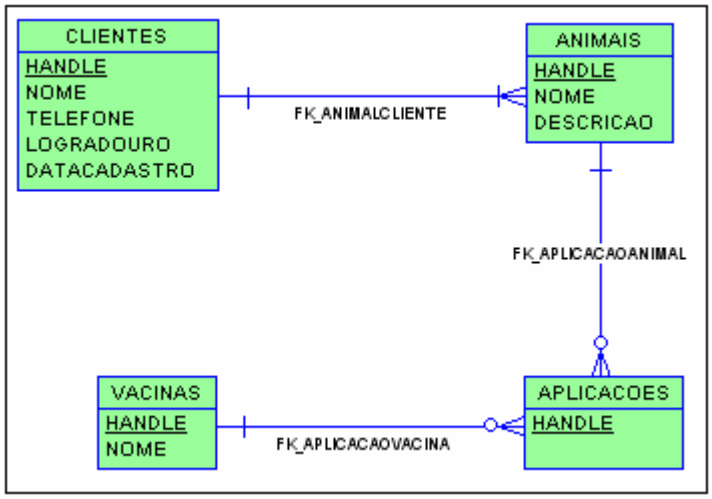

Figura 19 - Modelo conceitual na versão 1.0

A definição conceitual do esquema de dados da versão 1.0 é demonstrada na figura 19. A tabela 1 exibe o histórico da evolução do esquema de dados (*log* das alterações) para a versão 1.0. Esta tabela de alterações está representando logicamente cada alteração aplicada à base.

Para este sistema foi definido que para chaves primárias das tabelas é utilizado o campo *handle* criado automaticamente pela ferramenta. Este campo é criado juntamente com a tabela, sendo gerado somente um registro de *log* para esta alteração.

Esta primeira versão do sistema foi adquirida por duas clínicas veterinárias, que serão aqui denominadas de Casa dos Animais e Bicho Feliz. Na implantação do sistema, estes

clientes foram cadastrados na ferramenta para o controle de suas bases de produção. Foi então gerado um *script* de exportação com as alterações desta versão e atualizado os registros de alterações exportadas a estes dois clientes informando que as suas bases encontram-se nesta versão em questão. Este *script* executado pela ferramenta na base dos clientes (atualização), alterou seu esquema de dados tornando-as idênticas à base de desenvolvimento.

Tabela 1 - Histórico da evolução do esquema de dados na versão 1.0

Alteração	Descrição	Cliente	Versão
1	Criação da tabela CLIENTES		1.0
2	Criação do campo NOME na tabela CLIENTES		1.0
3	Criação do campo TELEFONE na tabela CLIENTES		1.0
4	Criação do campo LOGRADOURO na tabela CLIENTES		1.0
5	Criação do campo DATACADASTRO na tabela CLIENTES		1.0
6	Criação da tabela ANIMAIS		1.0
7	Criação do campo NOME na tabela ANIMAIS		1.0
8	Criação do campo DESCRICAO na tabela ANIMAIS		1.0
9	Criação do campo CLIENTE na tabela ANIMAIS		1.0
10	Criação da tabela VACINAS		1.0
11	Criação do campo NOME na tabela VACINAS		1.0
12	Criação da tabela APLICACOES		1.0
13	Criação do campo VACINA na tabela APLICACOES		1.0
14	Criação do campo ANIMAL na tabela APLICACOES		1.0

Após um certo período utilizando o sistema em sua clínica, o cliente Casa dos Animais passou a ter a necessidade de controlar qual de seus funcionários realizava a aplicação das vacinas em cada animal. Sendo este um desenvolvimento que teria a necessidade de ser implementado somente neste cliente, foi alterada a base de desenvolvimento sendo estas alterações definidas como específicas para este cliente. Nesta mesma situação o cliente Bicho Feliz necessitava que fossem armazenadas as idades dos animais em seus cadastros.

Por questões de desenvolvimento da aplicação, o campo de armazenamento do telefone no cadastro do cliente, foi alterado do tipo número para *string*. Sendo esta alteração a ser aplicada a todos os clientes.

Com estas modificações implementadas na base de desenvolvimento, foi denominada a situação do esquema de dados corrente como versão 2.0. Foram gerados novos *scripts* de exportação para cada um dos clientes, estes gerados através da ferramenta sendo um para cada cliente e contendo somente com as alterações conforme suas necessidades. Aplicadas estas alterações às bases de produção dos clientes, os esquemas conceituais das bases ficaram como

demonstrado na figura 20. A tabela 2 representa o histórico da evolução da base de desenvolvimento.

Na versão 2.0, há a diferença nos esquemas de dados dos clientes onde as definições específicas geraram variações de acordo com cada implementação. O cliente Casa dos Animais teve a entidade “Funcionários” criada, mas não o campo “Idade” como teve o cliente Bicho Feliz, as diferenças são ressaltadas na figura 20 pelos círculos vermelhos.

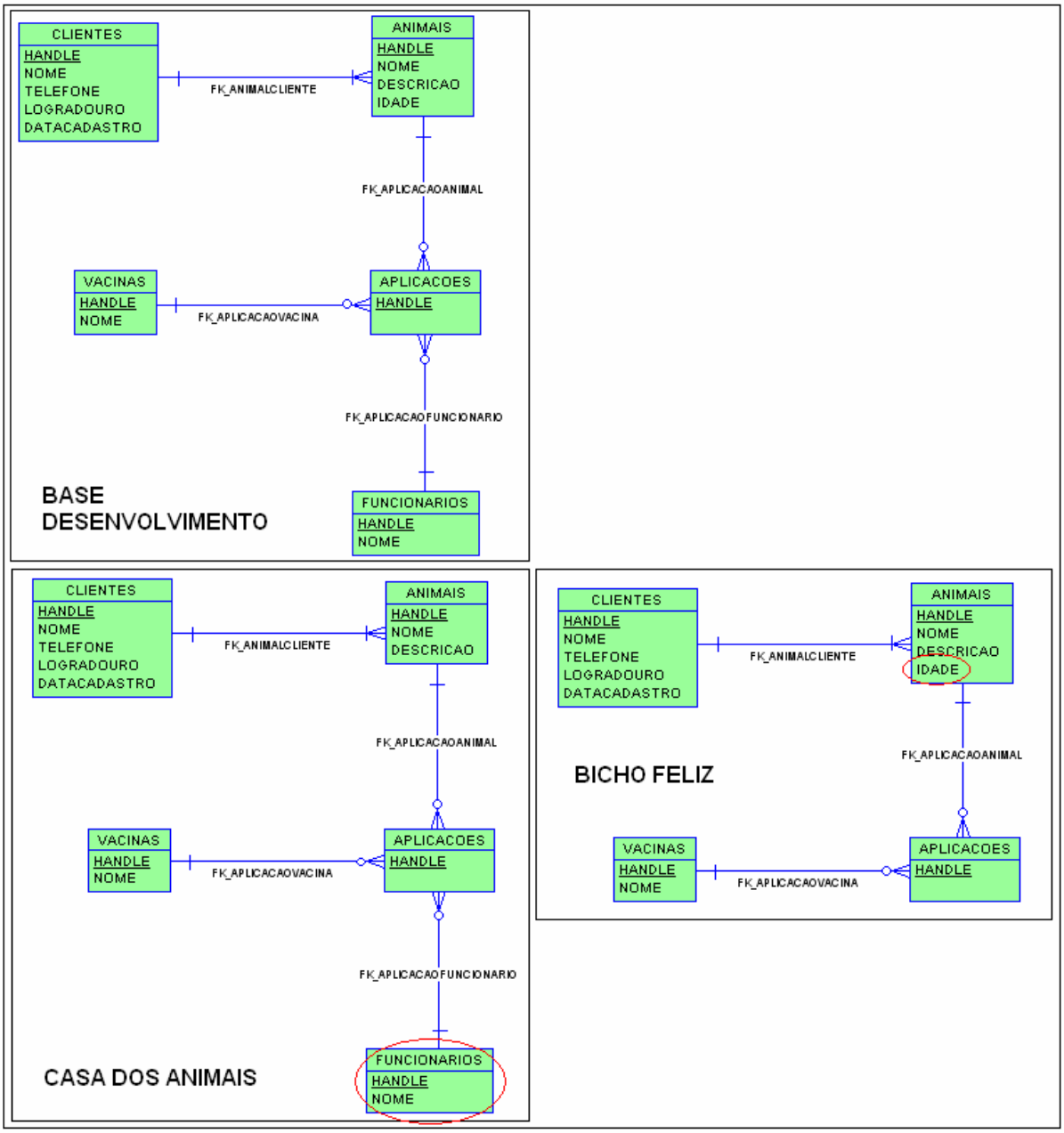

Figura 20 - Modelos conceituais na versão 2.0

Tabela 2 - Historio da evolução do esquema de dados na versão 2.0

Alteração	Descrição	Cliente	Versão
1	Criação da tabela CLIENTES		1.0
2	Criação do campo NOME na tabela CLIENTES		1.0
3	Criação do campo TELEFONE na tabela CLIENTES		1.0
4	Criação do campo LOGRADOURO na tabela CLIENTES		1.0
5	Criação do campo DATACADASTRO na tabela CLIENTES		1.0
6	Criação da tabela ANIMAIS		1.0
7	Criação do campo NOME na tabela ANIMAIS		1.0
8	Criação do campo DESCRICAO na tabela ANIMAIS		1.0
9	Criação do campo CLIENTE na tabela ANIMAIS		1.0
10	Criação da tabela VACINAS		1.0
11	Criação do campo NOME na tabela VACINAS		1.0
12	Criação da tabela APLICACOES		1.0
13	Criação do campo VACINA na tabela APLICACOES		1.0
14	Criação do campo ANIMAL na tabela APLICACOES		1.0
15	Criação da tabela FUNCIONARIOS	1	2.0
16	Criação do campo NOME na tabela FUNCIONARIOS	1	2.0
17	Criação do campo FUNCIONARIO na tabela APLICACOES	1	2.0
18	Criação do campo IDADE na tabela ANIMAIS	2	2.0
19	Alteração do campo TELEFONE na tabela CLIENTES		2.0

O campo cliente informado na tabela 2, tem como conteúdo “1” referenciando o cliente Casa dos Animais e como “2” referenciando o cliente Bicho Feliz.

Com a emergente expansão do produto no mercado, um novo cliente foi conquistado, aqui denominado de PetHouse. Para este cliente o sistema foi implantado na versão 2.0.

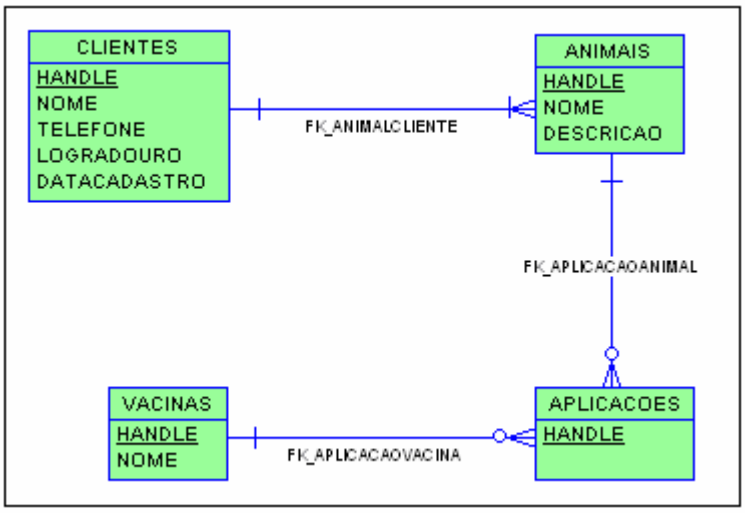

Figura 21 - Modelo conceitual do esquema de dados do cliente PetHouse

Gerado um *script* com o conjunto de alterações até a versão 2.0 para este cliente, e aplicado na sua base de produção, sua estrutura da é demonstrada na figura 21.

Novas alterações na base de desenvolvimento podem ser aplicadas nas bases de produção conforme a necessidade de cada cliente, e com o armazenamento das alterações já exportadas para cada cliente, é possível ter o gerenciamento da situação no qual cada base de produção se encontra.

7 CONCLUSÕES

Para o desenvolvimento deste trabalho, buscaram-se informações sobre os conceitos básicos de bancos de dados, a linguagem SQL e técnicas de versões de bases de dados. Possibilitou a aquisição de conhecimentos sobre o armazenamento e estrutura do banco SQL Server, como realizar a implantação física do esquema de dados através de comandos SQL e modos de gerenciamento de bases de dados.

Através dos testes realizados no protótipo desenvolvido, foi demonstrada a obtenção de uma interface para manipulação de bases de dados, que possibilitou as modificações serem realizadas de forma simples. O controle de versões obtido neste desenvolvimento permitiu o gerenciamento da evolução do esquema de dados, assim como a sua propagação nas bases de produções.

O Delphi mostrou-se adequado para o desenvolvimento deste trabalho. O uso de seus componentes, sem o conhecimento anterior destes à data do desenvolvimento, demonstrou a facilidade do Delphi para o desenvolvimento de aplicações que utilizam banco de dados.

O banco de dados SQL Server foi escolhido por ter uma boa aceitação no mercado, por possuir bibliografia disponíveis e pelo fato do acadêmico utiliza-lo no seu ambiente de trabalho profissional.

Com a realização deste trabalho tem-se o intuito de consolidar e expandir os conhecimentos adquiridos na formação profissional e acadêmica.

7.1 LIMITAÇÕES

Sendo um protótipo, algumas limitações foram encontradas:

a) a ferramenta possui a capacidade de reconhecer a estrutura de uma base de dados somente das modificações implantadas através da ferramenta;
b) não foram implementados todos os tipos de dados do banco SQL Server;
c) campos de tipo *blob* foram definidos com o tamanho fixo de 128;
d) para múltiplos usuários alterando a mesma base de dados simultaneamente, não é tido um controle mais aprimorado destas alterações, como por exemplo bloquear uma tabela enquanto esta sendo alterada por um usuário. Também não é realizada

uma transação do DBExpress para cada usuário conectado à base, sendo possível realizar uma transação por vez na ferramenta.

7.2 SUGESTÕES DE TRABALHOS FUTUROS

Seguem como sugestões para extensões deste trabalho:

a) estender a utilização desta ferramenta para os bancos de dados suportados pelo DBExpress;
b) permitir gerar *scripts* de exportação de conteúdo das entidades assim como da estrutura da base;
c) elaborar uma "varredura" da estrutura de uma base de dados, importando as definições para a estrutura da ferramenta, e assim, possibilitando-a de manipular uma base com um esquema de dados já implantado;
d) permitir de serem criados e removidos constraints e índices, de tabelas em particular ou de todas, para proporcionar ao projetista um maior gerenciamento da base de dados;
e) com a estrutura das tabelas definida na ferramenta, tornar possível serem gerados automaticamente formulários do Delphi com o tratamento do cadastramento de registros na tabela envolvida. Estes formulários possíveis de serem importados no Delphi e utilizados em uma aplicação em desenvolvimento.

REFERÊNCIAS BIBLIOGRÁFICAS

ALVES, William Pereira. **Deplhi 7:** aplicações avançadas de banco de dados. São Paulo: Érica, 2003.

BARBIERI, Carlos. **Modelagem de dados**. Rio de Janeiro: Infobook, 1994.

BATTISTI, Júlio. **SQL server 2000 administração e desenvolvimento:** curso completo. São Paulo: Axcel Books, 2001.

CANTU, Marco. **Mastering Delphi 7**. London: Sybex, 2003.

BJORNERSTEDT, Anders; HULTEN, Christer. **Version control in a object-oriented architecture**. Addison-Wesley, 1989.

CAMOLESI JR., Luís; TRAINA JR., Caetano. **Tratamento de múltiplos aspectos de projeto através de um gerenciador de esquema de dados para modelos orientados a objetos**, In: Simpósio Brasileiro de Banco de Dados, 8, 1993, Campina Grande. **Anais**... p. 283-296.

CAMOLESI JR., Luís. **Um modelo de versões apoiado em objetos compostos para utilização em instâncias e esquemas de bases de dados orientados a objetos**. 1996. 179 f. Tese (Doutorado) - Departamento de Física e Informática, Universidade de São Paulo, São Carlos. Disponível em: <http://www.unimep.br/~lcamoles/Dir-Publicacoes/Tese.pdf>. Acesso em 05/2004.

CERÍCOLA, Vincent Oswald. **Oracle**: Banco de dados relacional e distribuído. São Paulo: Makron Books, 1995.

FERNANDES, Lucia. **Oracle para desenvolvedores**: Oracle developer, Oracle 8-8i, curso completo. Rio de Janeiro: Axcel Books, 2000.

HURSCH, Carolyn J.; HURSCH, Jack L. **Linguagem de consulta estruturada SQL.** Rio de Janeiro: Livros Técnicos e Científicos, 1990.

LEAO, Renata de Oliveira; SILVA, João Carlos da. **SQL server 2000:** estrutura e implementação de sistemas de banco de dados. São Paulo: Érica, 2002.

MACHADO, Felipe Nery Rodrigues; ABREU, Maurício Pereira de. **Projeto de banco de dados**: Uma visão prática. São Paulo: Érica, 1996.

RAMALHO, Jose Antônio. **Oracle 8i**. São Paulo: Berkeley Brasil, 1999..

RIBEIRO, Ivanilson Lima. **Delphi 6 com SQL server 2000.** São Paulo: Érica, 2001.

SILBERSCHATZ, Avi; STONEBRAKER, Mike; ULLMAN, Jeff. **DataBase research**: Achievements and opportunities into the 21st century. Sigmod Record, 1996.

SWAN, Tom. **Delphi**: Bíblia do programador. São Paulo: Berkeley Brasil, 1996.

TRESH, Markus; SCHOLL, Mark. **Schema transformation without database reorganization**. Sigmod Record, 1993.

MICROSOFT. **Microsoft SQL Server 2000 -** Database design. New York: Element K, 2001.

8 APÊNDICE I - DICIONÁRIO DE DADOS

DBC_ALTERACOESCLIENTES

Nome	Código	Tipo	P	M
HANDLE	HANDLE	bigint	Sim	Sim
CLIENTE	CLIENTE	bigint	Não	Sim
VERSAO	VERSAO	bigint	Não	Sim
LOG	LOG	bigint	Não	Sim

DBC_CAMPOS

Nome	Código	Tipo	P	M
HANDLE	HANDLE	bigint	Sim	Sim
TABELA	TABELA	bigint	Não	Sim
NOME	NOME	varchar(60)	Não	Sim
DESCRICAO	DESCRICAO	varchar(60)	Não	Sim
TIPO	TIPO	tinyint	Não	Sim
TIPOSECUNDARIO	TIPOSECUNDARIO	tinyint	Não	Não
TAMANHO	TAMANHO	varchar(10)	Não	Não
DECIMAIS	DECIMAIS	varchar(3)	Não	Não
NAONULO	NAONULO	varchar(1)	Não	Não
TABELAPAI	TABELAPAI	int	Não	Não
SEMINTEGRIDADE	SEMINTEGRIDADE	varchar(1)	Não	Não
INDICE	INDICE	int	Não	Não
CLIENTE	CLIENTE	bigint	Não	Não

DBC_CLIENTES

Nome	Código	Tipo	P	M
HANDLE	HANDLE	bigint	Sim	Sim
NOMECLIENTE	NOMECLIENTE	char(80)	Não	Sim
VERSAO	VERSAO	int	Não	Sim

DBC_CONSTRAINTS

Nome	Código	Tipo	P	M
HANDLE	HANDLE	bigint	Sim	Sim
CAMPO	CAMPO	bigint	Não	Sim
TABELA	TABELA	bigint	Não	Sim
NOME	NOME	varchar(20)	Não	Sim

DBC_INDICES

Nome	Código	Tipo	P	M
HANDLE	HANDLE	bigint	Sim	Sim
TABELA	TABELA	int	Não	Sim
NOME	NOME	varchar(40)	Não	Sim
DESCRICAO	DESCRICAO	varchar(60)	Não	Não
CLIENTE	CLIENTE	int	Não	Não

DBC_LOG

Nome	Código	Tipo	P	M
HANDLE	HANDLE	BIGINT	Sim	Sim
DATA	DATA	datetime	Não	Sim
SQLBASE	SQLBASE	text	Não	Sim
SQLDBC	SQLDBC	text	Não	Sim
SQLCONSTRAINT	SQLCONSTRAINT	text	Não	Não
SQLCONSTRAINT2	SQLCONSTRAINT2	text	Não	Não
SQLINDICE	SQLINDICE	text	Não	Não
TIPO	TIPO	varchar(1)	Não	Sim
USUARIO	USUARIO	varchar(20)	Não	Sim
VERSAO	VERSAO	int	Não	Sim
CLIENTE	CLIENTE	bigint	Não	Não

DBC_TABELAS

Nome	Código	Tipo	P	M
HANDLE	HANDLE	bigint	Sim	Sim
NOME	NOME	varchar(30)	Não	Sim
DESCRICAO	DESCRICAO	varchar(60)	Não	Sim
CLIENTE	CLIENTE	bigint	Não	Não

DBC_VERSOES

Nome	Código	Tipo	P	M
HANDLE	HANDLE	int	Sim	Sim
VERSAO	VERSAO	char(20)	Não	Sim
FECHADA	FECHADA	char(1)	Não	Sim

9 APÊNDICE II – CRIAÇÃO DE UMA BASE DE DADOS NO SQL SERVER 2000.

Neste apêndice será abordada uma sintática referência de como criar uma base de dados no SQL Server 2000, para que a ferramenta desenvolvida neste trabalho possa ser utilizada. Para a instalação do SQL Server 2000 e detalhamento da criação da base de dados, é aconselhado a consulta de uma referência bibliográfica específica ou manual do próprio banco, para o detalhamento das características e funcionalidades deste banco de dados.

Será aqui considerado o banco de dados já instalado, para ser dada uma atenção voltada na criação da base e de usuários, omitindo configurações do banco não necessárias para este projeto.

O primeiro passo para a criação da base é confirmar se o serviço do SQL Server está ativo, gerenciado pelo utilitário Service Manager do SQL Server. O servidor no qual o banco está instalado e o serviço SQL Server devem estar informados, com o status de *running* (ativo), exemplificado na figura 22.

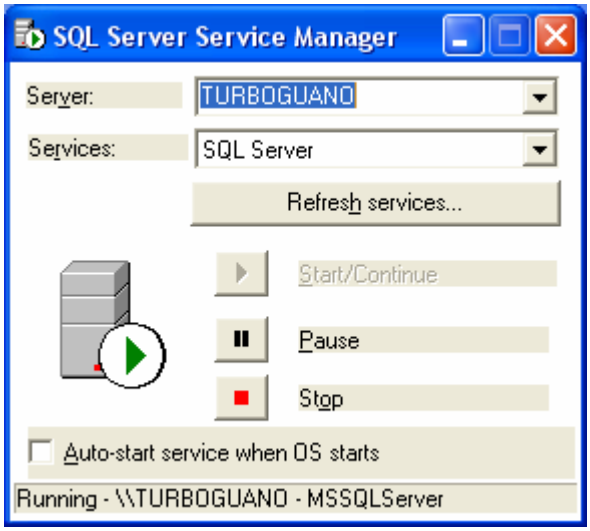

Figura 22 - Gerenciador de serviço do SQL Server

A ferramenta Enterprise Manager do SQL Server possibilita o gerenciamento do banco de dados, como criação/manutenção de bases e usuários, entre outras funcionalidades. Com o serviço do banco ativo, na pasta Databases do Enterprise Manager, estão listadas as bases existentes no banco (Figura 23).

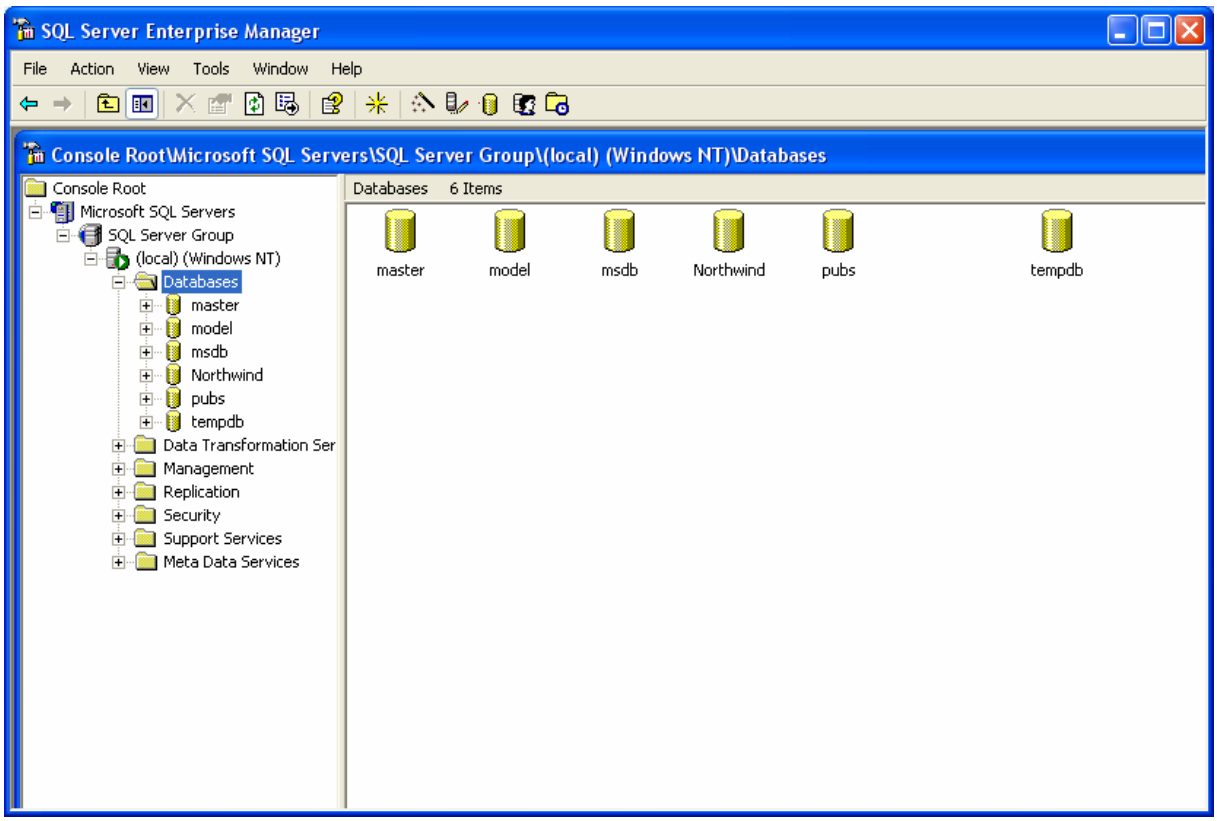

Figura 23 - SQL Server Enterprise Manager

Para criar uma nova base, deve-se selecionar a pasta Databases, e com o botão direito do mouse selecionar a opção “New Database”, onde o formulário exibido o formulário da figura 24. Na página “General” do formulário de propriedades de criação de bases (Figura 24 ,deve ser informado o nome da base a ser criada (aqui exemplifica de “TCC”), isto é o básico necessário para que a base seja criada. Outras características da base podem ser informadas neste formulário caso desejado uma configuração mais definida de acordo com as necessidades da sua utilização.

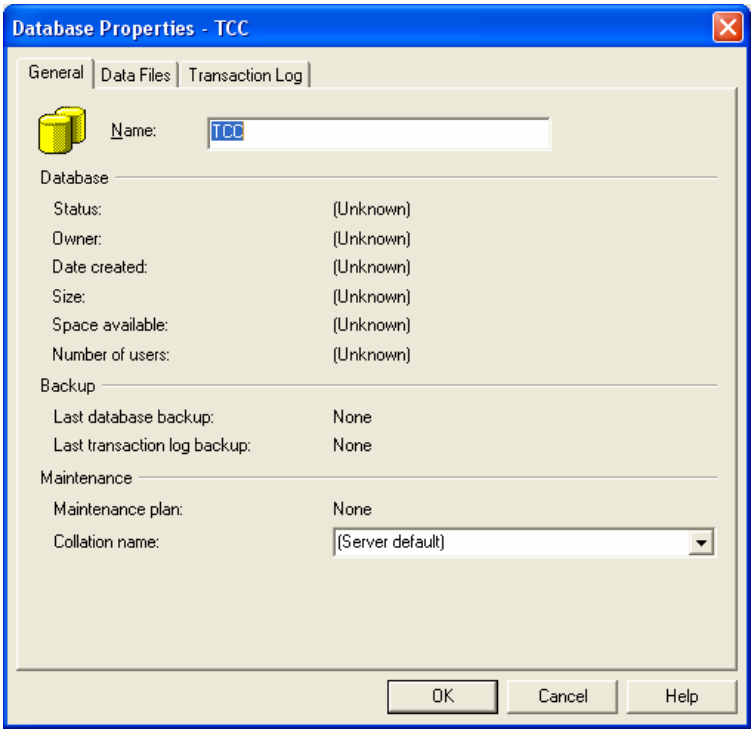

Figura 24 - Formulário de propriedades de uma nova base de dados

É necessário ainda a fazer a definição dos usuários que terão permissão de acesso à base criada, através do ícone indicado na figura 25. Neste ícone o formulário de criação de usuários é exibido (Figura 26).

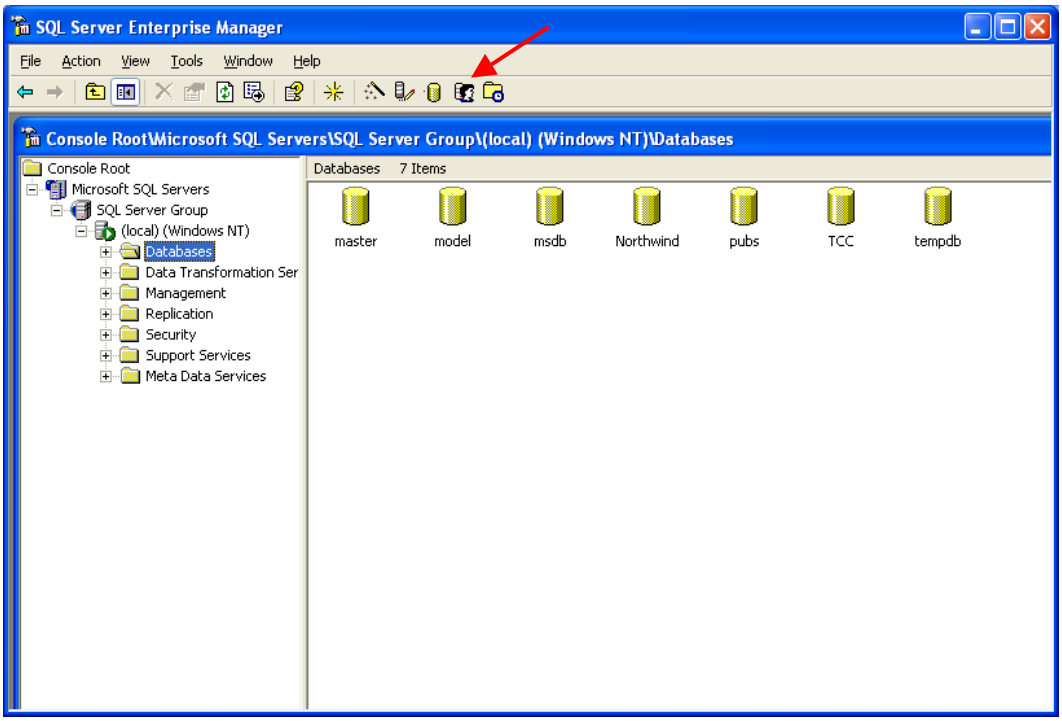

Figura 25 - Definição de usuários

Para definir um novo usuário, deve ser informado no campo “Name” do formulário exibido na figura 26, o nome de *login* para o acesso à base. Aconselha-se selecionar a opção “SQL Server Authentication” para definir a senha de acesso deste usuário. Na página “Database Access” define-se quais as bases este usuário terá acesso e as permissões que ele terá para cada base selecionada.

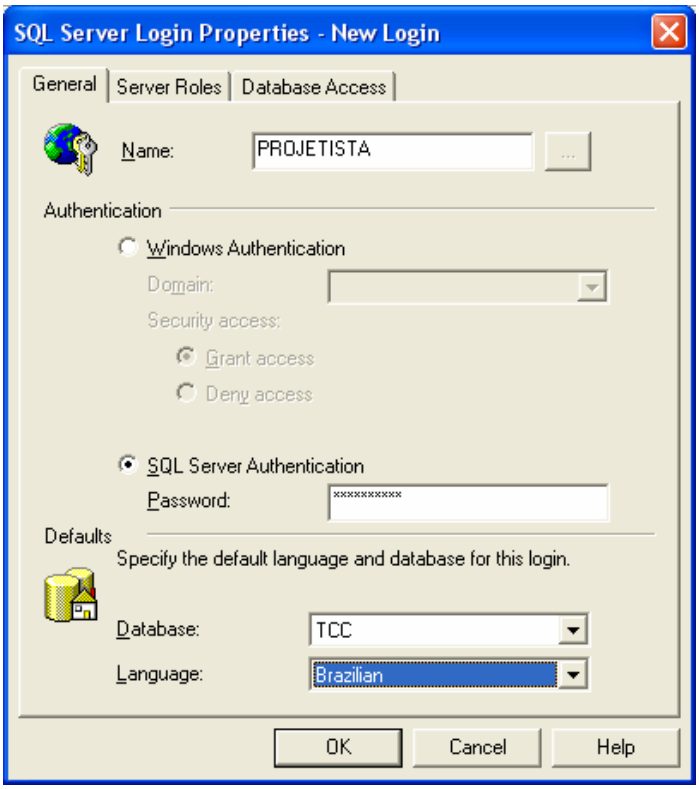

Figura 26 - Criação de novos usuários

Com a execução dos procedimentos demonstrados acima, é possível a utilização da ferramenta proposta neste trabalho no banco de dados SQL Server 2000.