WebModeler: uma ferramenta CASE para modelagem de banco de dados relacional na web

Alexander R. Valdameri[1], Juarez Bachmann[1]

[1]Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

arv@furb.br, juarez.bachmann@gmail.com

Resumo. *Este trabalho apresenta um aplicativo para definição do modelo lógico nos projetos de bancos de dados relacionais, desenvolvido para o ambiente web utilizando a plataforma Java, a arquitetura MVC e alguns outros padrões de projeto, como DAO e Command. Para tornar a interface do sistema mais amigável ao usuário, foi utilizado o conjunto de técnicas conhecido por Ajax, disponibilizando recursos semelhantes aos de aplicativos desktop, o que permite inclusive que sejam criados diagramas para modelar as tabelas, colunas e relacionamentos de forma gráfica.*

1. Introdução

A variedade de ferramentas CASE existentes no mercado com o propósito de auxiliar na modelagem de bancos de dados relacionais é grande, sendo que a maioria delas têm características em comum, como a possibilidade de modelar os dados de forma gráfica (desenhando tabelas e relacionamentos) e salvar os modelos criados em arquivos binários, que podem ser enviados a outras pessoas e utilizados por elas, desde que estas utilizem a mesma ferramenta. Outra característica marcante presente nestes softwares é dificuldade de manutenção sobre os mesmos, visto que são aplicações *desktop.* Além do custo para aquisição e instalação, têm a manutenção, que gera certa dificuldade para atualização caso estejam instaladas em inúmeras máquinas dentro de uma empresa, pois a atualização precisará ser feita em cada uma delas.

Uma alternativa interessante para amenizar eventuais problemas decorrentes das características dos aplicativos *desktop* é a internet, que vem passando por transformações significativas nos últimos anos. Neste momento, a web está virando uma plataforma: todo tipo de aplicativo é executado no *browser*. Inicialmente, foram os serviços de e-mail que saíram do *desktop* e foram disponibilizados na grande rede mundial de computadores. Agora, já há editores de textos, planilhas, agendas, diários, álbuns de fotos, enciclopédias e muitos outros serviços como estes. Além disso, diversos softwares corporativos são executados nos navegadores, como os portais de compras, por exemplo.

Diante deste contexto, pode-se dizer que há uma forte tendência em transformar os aplicativos *desktop* em ferramentas para o ambiente web e os softwares de modelagem de banco de dados também podem ser enquadrados nesta situação. Sendo assim, o objetivo do presente artigo é apresentar o desenvolvimento e operacionalidade de um aplicativo web (chamado WebModeler) que permite a definição de modelos lógicos para projetos de banco de dados relacional.

2. Desenvolvimento da ferramenta

A ferramenta desenvolvida implementa os casos de uso exibidos na Figura 1.

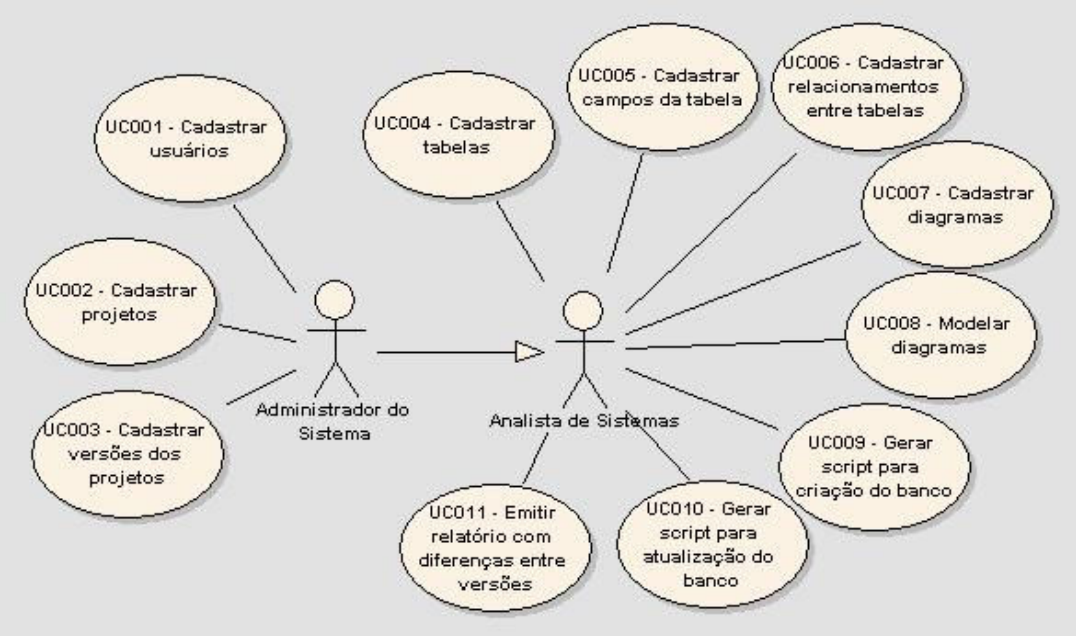

Figura 1. Casos de uso do sistema

Desta forma, as funcionalidades presentes neste software são:

- Cadastramento dos usuários que poderão acessar o sistema.
- Cadastramento dos projetos controlados.
- Ligação entre projetos e usuários, definindo permissões de acesso.
- Controle das versões de cada projeto, a fim de manter o registro histórico das alterações feitas no projeto com o passar do tempo.
- Cadastramento das tabelas existentes em cada versão, assim como suas colunas e relacionamentos. Para as colunas, é possível informar um tipo de dado, bem como definir se a mesma é parte da chave primária da tabela ou não. É possível ainda efetuar este cadastramento graficamente, na forma de diagramas.
- Geração de *scripts* com os comandos SQL para a criação do banco de dados nos SGBD Oracle e MSSqlServer e também para a atualização do banco de dados nestes SGBD com base na comparação entre duas versões de um projeto.
- Emissão de um relatório com a descrição das diferenças existentes entre uma versão e outra de um projeto, identificando tabelas e colunas que tenham sido incluídas, alteradas ou excluídas.

Nas seções 2.1, 2.2 e 2.3 há um detalhamento das técnicas e tecnologias mais relevantes empregadas no desenvolvimento deste aplicativo. Já a seção 2.4 descreve o processo de implementação da ferramenta.

2.1. Ambiente de desenvolvimento

Para a implementação foi utilizada a linguagem Java, de acordo com a especificação JEE (até pouco tempo chamada J2EE), sendo que o ambiente de desenvolvimento utilizado foi o Eclipse, na versão 3.2. O servidor de aplicações utilizado para executar o software foi o Apache Tomcat (versão 5.5) e o SGBD utilizado foi o MySql.

2.2. Padrões de projeto de software

Bond et al. (2003) descreve os padrões dentro da área de software como uma forma de capturar parte da experiência dos arquitetos e projetistas de sucesso, para que ela possa ser aplicada mais amplamente, ou seja, "um padrão é uma idéia reutilizável sobre como resolver um problema em particular, encontrado no domínio da arquitetura ou projeto" [Bond et al., 2003].

Os padrões podem ser divididos em tipos, sendo que segundo Bond et al. (2003), os mais comuns são os padrões de arquitetura (que definem o estilo geral do sistema, como a quantidade de camadas que a aplicação terá e o relacionamento entre elas) e os padrões de projeto (que se encontram no nível dos artefatos como classes e componentes). Além destes, também há os chamados padrões do JEE, que são um conjunto de padrões que têm sido identificados dentro das soluções com base no JEE, padrões gerais já identificados e utilizados anteriormente em outras plataformas [Bond et al., 2003].

A seguir, são descritos os padrões utilizados neste trabalho.

2.2.1. *Model-View-Controller* (MVC)

Gamma et al. (2000), descreve MVC como um padrão de arquitetura que divide o software em três tipos de objetos para aumentar a flexibilidade e a reutilização. O primeiro é o Modelo (*Model*), que consiste na camada onde ocorre o processamento das informações, onde os dados são gravados nos bancos de dados e recuperados dos mesmos, ou seja, é no Modelo que ficam as regras de negócio. O segundo tipo de objeto é a Vista ou Visão (*View*), cuja função é a de apresentar as informações ao usuário, isto é, este objeto é a interface do sistema. O terceiro e último tipo é o Controlador (*Controller*), que define como a interface reage às entradas do usuário. A Visão sempre deve refletir a situação atual do Modelo, sendo que o Controlador é responsável por controlar o fluxo entre as duas outras camadas.

2.2.2. *Data Access Object* (DAO)

Segundo Bond et al. (2003) o DAO é um padrão do JEE que visa criar objetos que encapsulem o acesso aos dados por trás de uma interface comum, que pode ser implementada de diferentes formas para diferentes fontes de dados.

Sullivan (2003) explica que os desenvolvedores JEE utilizam este padrão para separar a lógica de acesso a dados (baixo nível) da lógica das regras de negócio (alto nível). Com isso, de acordo com Bond et al. (2003), ficam amenizados dois problemas que ocorrem com freqüência nos sistemas onde o código de acesso a dados é mesclado com a lógica do negócio: dificuldade de manutenção e pouca flexibilidade. Isto ocorre porque ao utilizar este padrão, uma alteração no código de acesso a dados só precisará

ser feita no método específico da classe DAO em questão, não sendo necessário procurar e alterar inúmeros pontos do sistema que possuem os mesmos comandos SQL.

2.2.3. *Command*

Segundo Gamma et al. (2000) o padrão *Command* visa encapsular uma solicitação como um objeto, permitindo parametrizar clientes com diferentes solicitações, fazer o registro (log) de solicitações e também suportar operações que podem ser desfeitas.

Este padrão, também conhecido como *Action* ou *Transaction*, possibilita também estruturar um sistema em torno de operações de alto nível, estrutura esta muito comum em sistemas que suportam transações. Assim, pode-se dizer que o padrão *Command* fornece uma maneira de modelar transações [Gamma et al., 2000].

O princípio básico desse padrão é a criação de uma classe abstrata `Command`, com uma operação abstrata `execute()`, sendo que esta operação é implementada nas subclasses concretas de `Command`. Em um editor de textos hipotético, cada botão do menu que fosse acionado criaria uma instância de uma subclasse de `Command`, de acordo com sua funcionalidade. Um botão chamado Salvar Documento, por exemplo, instanciaria um objeto da classe `SalvarCommand` e em seguida dispararia o seu método `execute()`, sendo que o mesmo aconteceria para os outros botões e classes. Desta forma, cada operação é executada na sua totalidade por uma classe específica, facilitando a compreensão e manutenção do sistema.

2.2.4. *Front Controller*

Em Bond et al. (2003) o *Front Controller* é definido como um padrão JEE no qual um componente intercepta a requisição do usuário e direciona ou agrega valor a mesma.

A definição encontrada em Sun Microsystems (2002b) diz que este padrão consiste em definir um componente simples que seja responsável pelo processamento das requisições da aplicação. Este componente recebe a requisição, encaminha-a para o componente responsável por processá-la (outra classe ou *servlet*) e após o processamento seleciona a página que deve ser apresentada ao usuário. Além disso, este componente centraliza funções relativas à segurança e tratamento de erros, facilitando a manutenção do software [Sun Microsystems, 2002a].

2.3. Ajax

O termo Ajax foi criado por Jesse James Garret em fevereiro de 2005, como um acrônimo para *Asynchronous JavaScript and XML*, mas atualmente é usado para englobar todas as tecnologias que possibilitam ao navegador se comunicar com o servidor sem atualizar toda a página atual [Asleson e Schutta, 2006].

Segundo Asleson e Schutta (2006), o Ajax pode ser considerado como uma técnica e não uma tecnologia específica. Esta técnica agrega tecnologias como JavaScript, *eXtensive Markup Language* (XML), *Cascading Style Sheets* (CSS), *Document Object Model* (DOM) e o objeto do JavaScript chamado `XMLHttpRequest` para tornar as páginas web mais dinâmicas, disponibilizando recursos como criação de menus de contexto que aparecem ao clicar com o botão direito do *mouse*, possibilidade de deslizar controles (arrastar e soltar) e trocar informações com o servidor de forma assíncrona, sem a necessidade de atualizar toda a página ao

fazê-lo. Todas estas tecnologias são tratadas no *browser*, ou seja, "o Ajax é uma abordagem do lado cliente e pode interagir com a JEE, .NET, PHP, Ruby e scripts CGI – realmente não depende do servidor" [Asleson e Schutta, 2006], sendo que a tecnologia mais recente (e talvez mais importante) relacionada ao termo, é o objeto `XMLHttpRequest`, responsável pela comunicação assíncrona entre o navegador e o servidor web.

2.4. Processo de implementação da ferramenta WebModeler

A implementação do sistema ocorreu conforme os passos descritos nas seções seguintes.

2.4.1. Desenvolvimento da camada de persistência e recuperação dos dados

Nesta etapa foram desenvolvidas as classes que implementam o padrão de projeto DAO, ou seja, as classes responsáveis por gravar e recuperar as informações no banco de dados. Para cada tabela do banco de dados da aplicação foi criada uma classe DAO correspondente, com métodos para inserir, alterar, excluir e listar seus respectivos registros. A Tabela 1 apresenta os principais métodos criados na classe UsuarioDAO.

Tabela 1 – Principais métodos da classe UsuarioDAO

Método	Objetivo
carregar(int)	Retorna um objeto do tipo Usuario, com os dados do usuário cujo código foi passado como parâmetro.
excluir(int)	Exclui do banco de dados o usuário cujo código foi passado como parâmetro.
gravar(Usuario)	Recebe um objeto do tipo Usuario como parâmetro e insere/altera os dados do mesmo no banco.
listar()	Retorna uma lista de objetos do tipo Usuario com todos os usuários cadastrados no sistema.

2.4.2. Desenvolvimento do núcleo de processamento de requisições do sistema

A parte do sistema aqui nomeada como "núcleo de processamento de requisições" nada mais é do que a implementação do padrão *Front Controller* em conjunto com o padrão *Command*, além de ser a camada *Controller* da arquitetura MVC.

Seguindo o que foi exposto anteriormente acerca do padrão *Front Controller*, criou-se uma classe de mesmo nome, a qual consiste num *servlet* Java que, segundo o que foi definido no arquivo de configuração web.xml, é responsável por atender todas as requisições cuja *url* termine com ".do".

Para o tratamento das requisições foi empregado o padrão de projeto *Command*. Com isto, foi implementada uma super-classe com o mesmo nome deste padrão, a qual possui apenas um método denominado `executar()` onde a lógica de cada `Command` foi escrita, ou seja, é neste método que ocorre o processamento das requisições e é onde se encontram as regras de negócio do software. Esta classe foi estendida para implementar o atendimento de cada tipo de requisição existente em cada módulo do sistema. Para atender a requisição de exclusão de um usuário, por exemplo, o

`FrontController` recebe a requisição de *url* "usu_excluir.do", instancia um objeto da classe `ExcluirUsuarioCommand` e dispara seu método `executar()`, sendo este funcionamento idêntico para as demais requisições.

2.4.3. Implementação dos cadastros básicos do sistema

Concluído o desenvolvimento da camada de persistência e do núcleo de processamento de requisições, foram implementadas as rotinas relativas aos cadastros de projetos, versões, usuários, tabelas, colunas, relacionamentos e diagramas.

Nesta fase, o esforço foi concentrado no desenvolvimento das telas (páginas jsp) com os campos necessários para cada cadastro e também das respectivas classes `Command` responsáveis pela recuperação, inclusão, alteração e exclusão dos registros. As telas de cadastro se comunicam assincronamente com o servidor através do objeto `XMLHttpRequest`, isto é, sempre que um botão é acionado, um código JavaScript é executado para criar o objeto `XMLHttpRequest` e enviar a requisição ao servidor. Após o servidor processar a solicitação através de uma classe `Command`, o mesmo devolve para o *browser* um arquivo XML que é tratado por outro código JavaScript, atualizando somente os valores dos campos ao invés de atualizar a página inteira, como ocorre em aplicações web tradicionais.

2.4.4. Implementação do módulo de modelagem gráfica

A modelagem de um banco de dados no sistema de forma gráfica visa disponibilizar ao analista uma interface amigável, na qual seja possível visualizar facilmente a estrutura do banco que está sendo projetado. Essa interface exibe, basicamente, as tabelas cadastradas (com suas colunas) e os relacionamentos entre elas, sendo as tabelas representadas por meio de retângulos em cujo interior aparecem as colunas e seus respectivos tipos de dados. Já os relacionamentos são representados por linhas conectadas às duas tabelas relacionadas, com uma seta na extremidade ligada à mestre.

No desenvolvimento desta parte do sistema, o Ajax foi utilizado tanto ou mais do que nas telas de cadastro. Utilizando JavaScript, DOM, CSS e o objeto `XMLHttpRequest` foi possível criar recursos para exibir as tabelas e relacionamentos, criar menus de contexto (botão direito do *mouse*) e arrastar e soltar (*drag-and-drop*) as tabelas. As tabelas são "desenhadas" na tela através da criação de um objeto HTML do tipo DIV (*tag* da linguagem HTML), objeto este capaz de definir uma área do HTML que pode ser formatada diferentemente do restante da página, além de reconhecer eventos como clique do *mouse* ou acionamento do teclado. Para que seja possível arrastar e soltar as tabelas, basicamente foi necessário associar uma função JavaScript ao evento `onmousemove` da página, função esta que trata o momento em que o usuário move o *mouse*, reposicionando o DIV selecionado de acordo com a posição do *mouse.*

Para que o usuário possa informar os dados da tabela e das suas colunas, foram disponibilizadas duas formas de acesso à tela de cadastro de tabelas: através de um duplo-clique sobre o DIV ou através do menu de contexto que aparece ao clicar com o botão direito do *mouse* sobre o mesmo. Ambas as opções levam o usuário à tela de cadastro de tabelas (a mesma que já havia sido desenvolvida anteriormente) que é aberta em uma nova página do *browser*. Após informar os dados desejados, o usuário

aciona um botão que fecha esta tela e faz com que, através de uma outra função JavaScript, o DIV relativo à tabela em questão seja redesenhado para exibir corretamente o nome da tabela e os dados de suas colunas. Esta função que atualiza o DIV na tela utiliza o objeto `XMLHttpRequest` para obter as informações necessárias do servidor assincronamente, sem a necessidade de atualizar a página inteira, ou seja, apenas o DIV é atualizado.

Para cadastrar um relacionamento, foi incluída uma opção no menu de contexto das tabelas, a qual abre a tela de cadastro de relacionamentos em uma nova página do *browser*. Depois de feito o cadastro, ao pressionar o botão para adicionar o relacionamento no diagrama, a tela de cadastro é fechada e o relacionamento é desenhado na tela. Para formar a linha e a seta que representam um relacionamento, são criados inúmeros pequenos DIVs, cada um ocupando 2 *pixels* de altura e outros 2 de largura, sendo que foi implementado um algoritmo que, a partir das posições das tabelas mestre e filha no diagrama, cria estes inúmeros DIVs até conectar uma tabela à outra.

Por fim, foi implementada a rotina de gravação da modelagem gráfica, para que os dados das tabelas e relacionamentos adicionados no diagrama sejam salvos no banco de dados, onde também foi utilizado o objeto `XMLHttpRequest` para que a gravação seja feita assincronamente.

2.4.5. Implementação do módulo de geração de *scripts*

Após a modelagem há a possibilidade de gerar arquivos com os comandos SQL necessários para criar um banco de dados e também para atualizar um banco com base nas alterações feitas entre uma versão de um projeto e outra. Até o momento, o sistema faz a geração destes *scripts* apenas para os SGBD Oracle 10g e MSSqlServer 2000, podendo suportar outros bancos em versões futuras. Na tela do sistema onde são cadastradas as tabelas, o usuário informa tipos de dados próprios do sistema para as colunas, com nomenclaturas em Língua Portuguesa (Numérico, Texto Variável, Data, etc.) ao invés de tipos próprios de algum SGBD (Varchar, Number, Int, etc.), sendo que os tipos próprios do sistema e os tipos nativos dos SGBD foram relacionados conforme descrito na Tabela 2.

Tabela 2 – Relacionamento entre os tipos de dados

WebModeler	**Oracle**	**MSSqlServer**
Texto Variável	Varchar2	Varchar
Texto Fixo	Char	Char
Inteiro	Integer	Int
Inteiro Curto	Smallint	Smallint
Inteiro Longo	Integer	Bigint
Numérico	Decimal	Decimal
Data	Date	DateTime
Data/Hora	DateTime	DateTime
Binário	Blob	Binary

O relacionamento descrito na Tabela 2 foi elaborado com base nas informações de Oracle (2005) e Battisti (2001) e, uma vez definido este relacionamento, foram desenvolvidas as classes e rotinas para geração dos *scripts* de criação do banco de dados. Logo em seguida, foram implementadas as classes responsáveis por gerar os *scripts* de atualização de bancos de dados através da comparação entre duas versões de um projeto. Estas classes utilizam o relacionamento apresentado na Tabela 2 para gerar os comandos SQL para cada banco, sendo que é utilizada a sintaxe do SQL padrão e não é possível personalizar os tipos de dados a serem utilizados para cada banco ou a sintaxe dos comandos. Por fim, foram desenvolvidas as classes que geram um relatório indicando quais tabelas e colunas foram adicionadas, alteradas ou excluídas de uma versão para a outra. Tanto os *scripts* quanto o relatório são apresentados ao usuário no *browser* na forma de um arquivo PDF único.

3. Operacionalidade e principais telas da ferramenta

A Figura 2 exibe um diagrama que ilustra o fluxo de trabalho no WebModeler, mostrando a divisão das funcionalidades disponíveis para o Administrador e para o Analista bem como a seqüência em que cada funcionalidade deve ser executada.

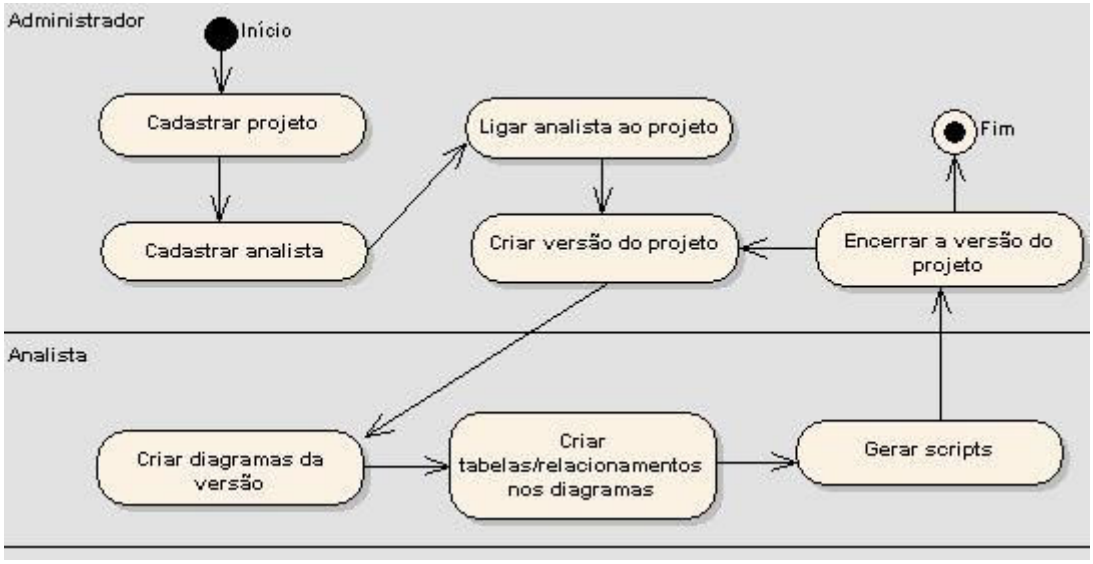

Figura 2. Fluxo de trabalho no WebModeler

A tela de cadastro de projetos é ilustrada na Figura 3.

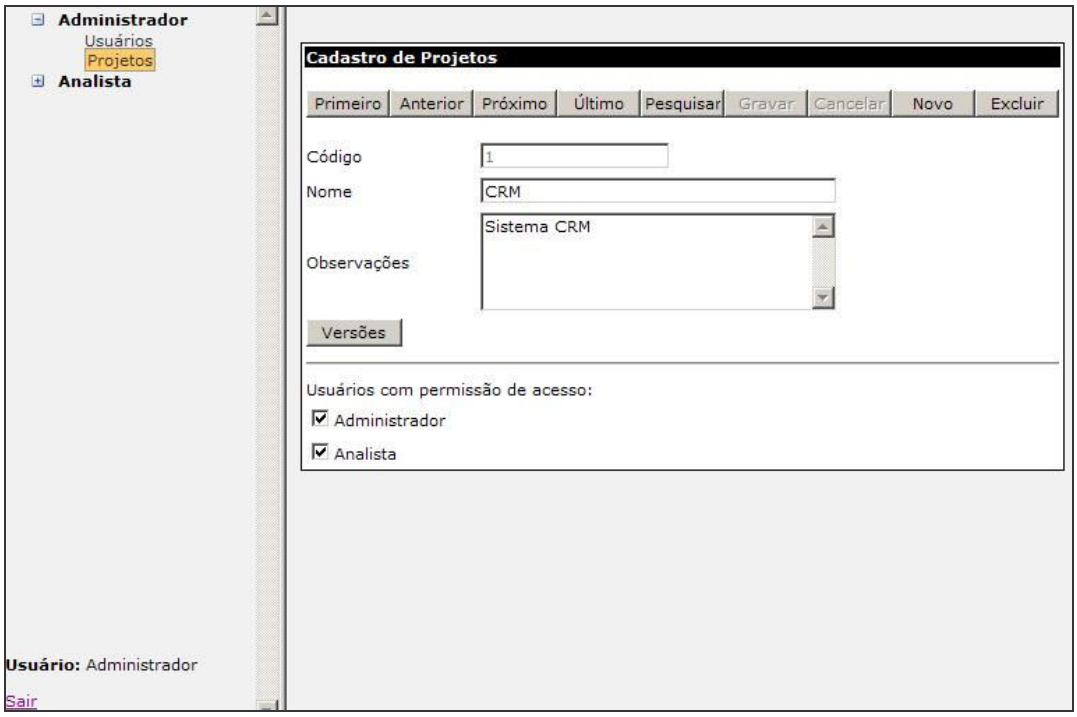

Figura 3. Tela de cadastro de projetos

Todas as outras telas de cadastro (versões dos projetos, usuários, diagramas, relacionamentos e tabelas) têm interface bastante semelhante à de cadastro de projetos e, por isso, nem todas serão exibidas neste artigo.

A Figura 4 demonstra a tela de cadastro de tabelas, sendo que na parte superior da mesma é informado o nome da tabela e na parte inferior são cadastradas as colunas.

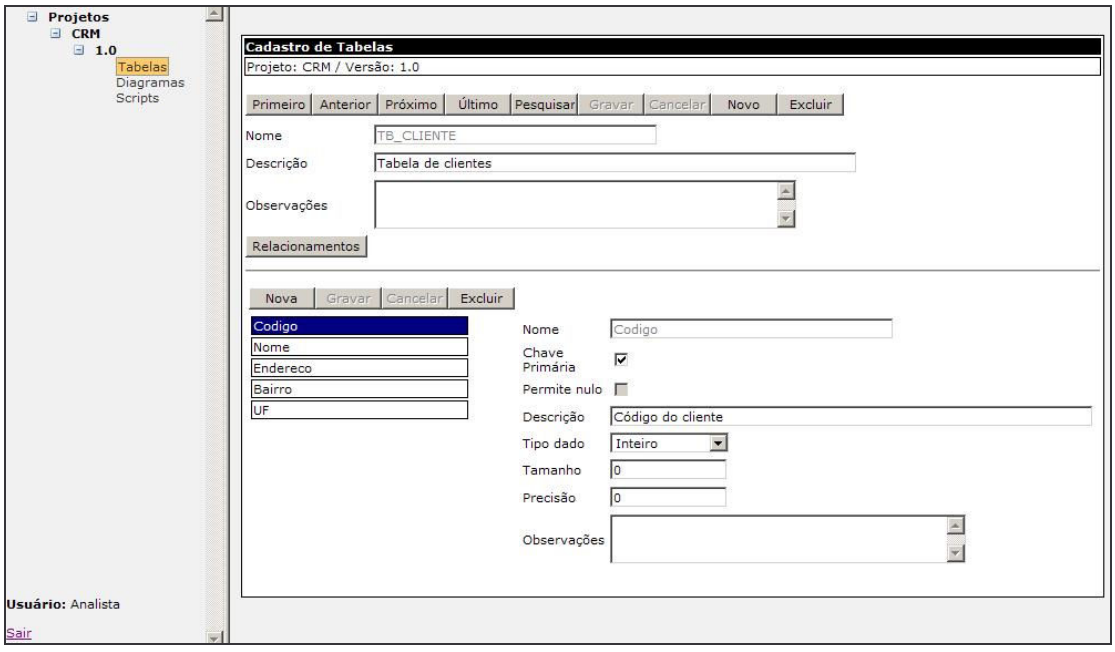

Figura 4. Tela de cadastro de tabelas

A Figura 5 exibe a tela de cadastro de relacionamentos entre as tabelas.

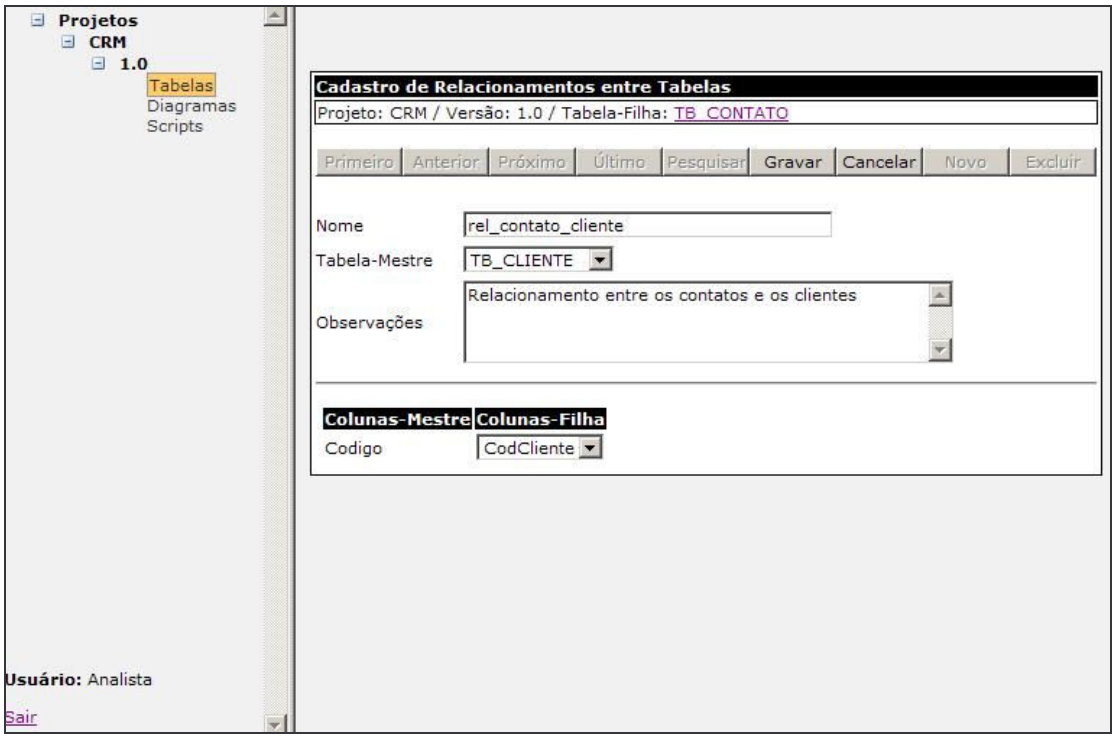

Figura 5. Tela de cadastro de relacionamentos entre tabelas

Já a Figura 6 mostra a tela de modelagem gráfica.

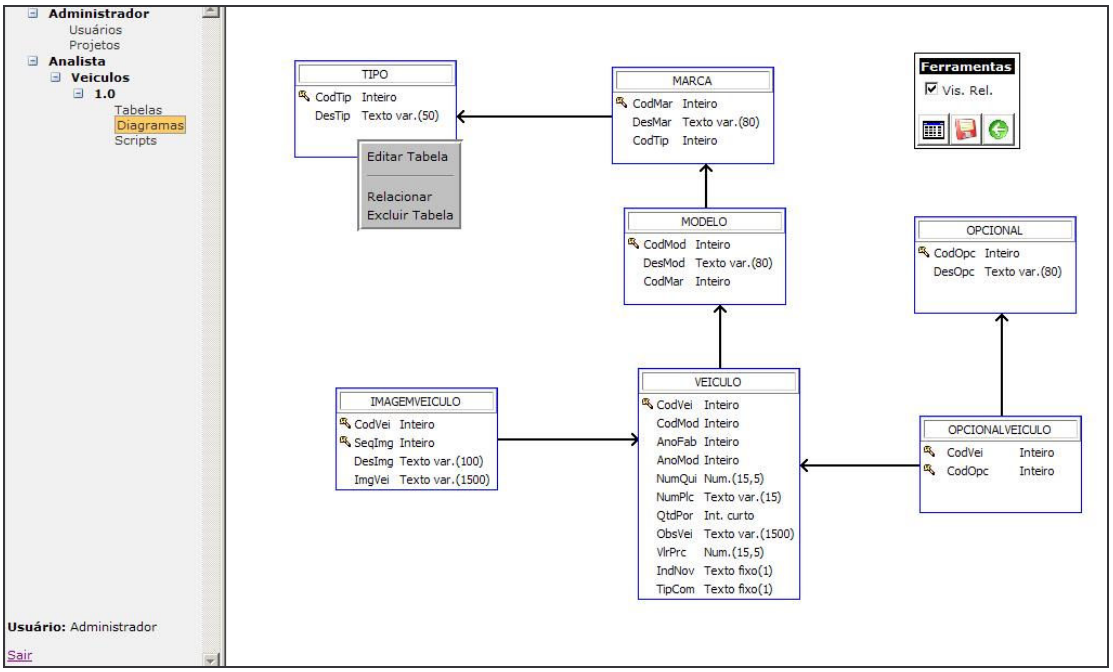

Figura 6. Tela de modelagem gráfica

Por fim, a Figura 7 ilustra a tela de geração de *scripts* e do relatório de diferenças entre versões de um projeto.

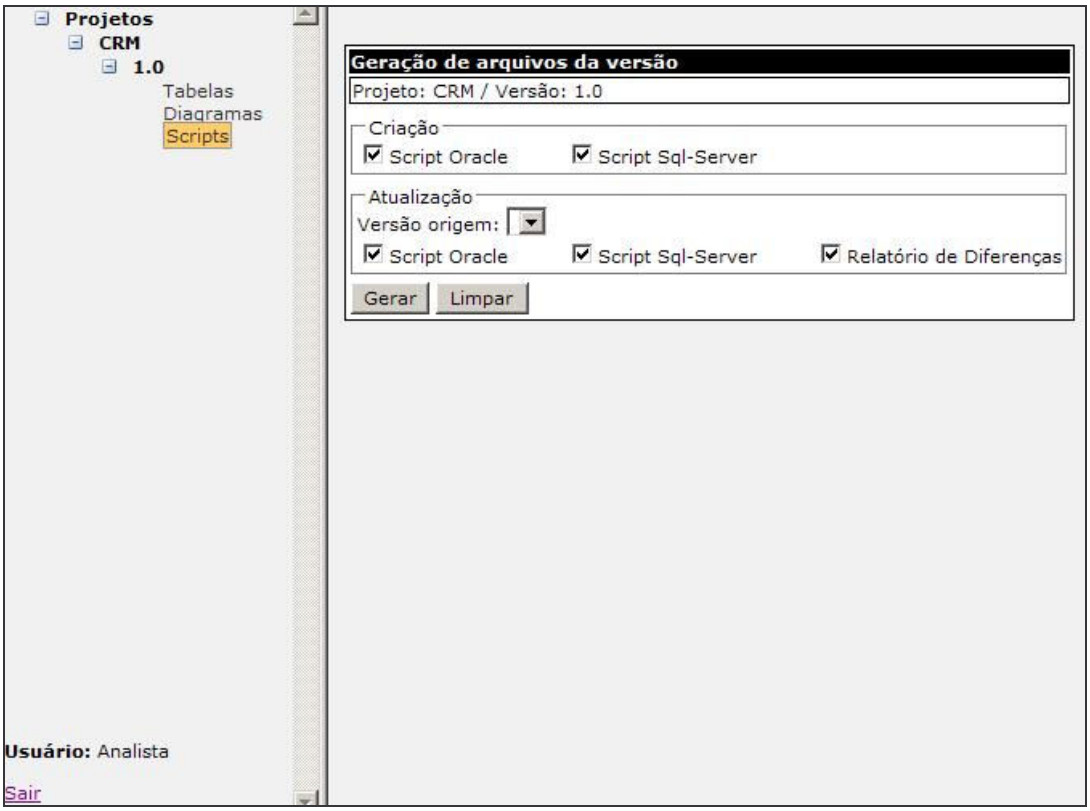

Figura 7. Tela de geração de *scripts*

4. Considerações finais

O software cujo desenvolvimento e operacionalidade foram apresentados neste artigo permite a modelagem de bancos de dados relacionais no ambiente web, possibilitando o cadastramento de projetos e suas versões, bem como das tabelas e relacionamentos existentes em cada versão. Essa modelagem pode ser feita inclusive de forma gráfica, como ocorre em diversas ferramentas CASE disponíveis no mercado, sendo que no software ora apresentado, as tabelas são modeladas definindo tipos de dados próprios do sistema, ao invés de tipos nativos do SGBD onde o banco será criado. Essa independência de algum SGBD específico pode ser considerada importante, principalmente porque diversas linguagens de programação modernas, como Delphi e Java, por exemplo, permitem a geração de programas cujo código-fonte é totalmente compatível com vários SGBDs relacionais, ou seja, não é necessária uma preocupação com o SGBD que será utilizado posteriormente e, com a ferramenta apresentada neste trabalho, também é possível criar o modelo do banco de dados sem esta preocupação. É possível ainda gerar *scripts* para a criação do banco de dados nos SGBD Oracle e MSSqlServer e, graças ao controle de versões, também é possível gerar *scripts* de atualização de bancos de dados.

Com relação às ferramentas CASE existentes no mercado, o software implementado tem algumas limitações, principalmente na questão da modelagem gráfica, pois não há recursos como impressão e *zoom* do diagrama, além de recursos para formatação das tabelas e relacionamentos. Muito da limitação da modelagem gráfica do WebModeler deve-se à atual falta de recursos dos *browsers* e do JavaScript em se tratando de criação de elementos gráficos (retas, figuras geométricas, etc.), sendo

que uma alternativa para amenizar estas limitações em versões futuras é a utilização da linguagem *Scalable Vector Graphics* (SVG), linguagem esta utilizada para criar recursos gráficos na web, mas sem suporte nativo no Internet Explorer. Em contrapartida às limitações, o WebModeler oferece vantagens como o acesso remoto (por tratar-se de um aplicativo web), controle de projetos e versões, cadastro de usuários e restrições de acesso por meio do relacionamento entre projetos e usuários.

Finalmente, com relação ao mercado para utilização da ferramenta ora apresentada, percebeu-se que além dos fins comerciais (empresas desenvolvedoras de software) a mesma pode ser utilizada para fins didáticos, em especial em disciplinas acadêmicas que visam prover conhecimentos básicos a cerca de bancos de dados relacionas, projeto de bancos de dados e modelagem de sistemas.

Referências

Asleson, R. e Schutta, N. T. (2006) "Fundamentos do Ajax", Rio de Janeiro: Alta Books.

Battisti, J. (2001) "SQL Server 2000: administração e desenvolvimento: curso completo", Rio de Janeiro: Acxel Books.

Bond, M. et al (2003) "Aprenda J2EE em 21 dias: com EJB, JSP, Servlets, JNDI, JDBC e XML", Tradução João Eduardo Nóbrega Tortell,. São Paulo: Pearson Education do Brasil.

Crane, D., Pascarello, E. e James, D. (2007) "Ajax em ação", Tradução Edson Furmankiewicz e Carlos Schafranski, São Paulo: Pearson Prentice Hall.

Gamma, E. et al. (2000), "Padrões de projeto: soluções reutilizáveis de software orientado a objetos", Tradução Luiz A. Meirelles Salgado, Porto Alegre: Bookman.

Oracle (2005) "Oracle Database SQL Reference 10g Release 2: Datatypes", Disponível em: <http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14200/sql_elements001.htm#i54330/>. Acesso em: 24 maio 2007.

Sullivan, S. (2003) "Advanced DAO programming", Disponível em: <http://www-128.ibm.com/developerworks/library/j-dao/>. Acesso em: 30 abr. 2007.

Sun Microsystems (2002a) "Core J2EE Patterns: Front Controller", Disponível em: <http://java.sun.com/blueprints/corej2eepatterns/Patterns/FrontController.html>**.** Acesso em: 24 abr. 2007.

Sun Microsystems (2002b) "Front Controller", Disponível em: <http://java.sun.com/blueprints/patterns/FrontController.html>**.** Acesso em: 24 abr. 2007.