MAC 499 - Trabalho de Formatura Supervisionado
Monografia

Projeto: Ricardo Guilherme Drizin
Orientador: Walter Figueiredo Mascarenhas

December 6, 2004

Contents

1 Objetivos do Projeto

1.1 Objetivos Pessoais

Creio que devo mencionar que, acima de tudo, o que me levou a idéia deste projeto foi um desejo pessoal de complementar os conhecimentos adquiridos ao longo do curso. O curso foi muito proveitoso e de ótimo nível teórico, no entanto como alguém que já está no mercado há alguns anos, eu sabia que o mercado de trabalho não era só isso.

Resolvi então iniciar um projeto que me propiciasse aprender algumas tecnologias que o curso do BCC não ensina e ao mesmo tempo aperfeiçoar alguns tópicos que ao longo do curso eu não cursei ou não dei a devida importância.

1.2 O Surgimento da idéia

A idéia do ERWon surgiu de um desejo que eu tinha de fazer algo relacionado a bancos de dados. Também queria aprender a linguagem C# e a plataforma .NET em geral.

Em uma primeira reunião com o meu orientador, o professor Walter Mascarenhas, eu esbocei algumas idéias sobre um utilitário para importar/exportar modelos de dados. O professor sugeriu então que eu fizesse um editor completo e me mostrou um programa que ele havia feito em C# em que era possivel manipular shapes na tela.

Foi a motivação que eu precisava - a linguagem C#, a idéia de que fazer uma interface gráfica não seria difícil (algo que até então eu temia), e um orientador que conhecia Microsoft .NET. Estava lançado então o desafio - eu aprenderia C#/.NET, um pouco sobre programação para Windows, programação gráfica em geral e ainda estudaria um pouco sobre bancos de dados.

1.3 ERWon - O que é? O que faz?

ERWon é um editor gráfico de modelos ER (entidade-relacionamento), feito na linguagem C# (Microsoft.NET Framework) e multiplataforma (nativamente para Windows e portável para Linux através do projeto DotGNU).

O editor é capaz de importar e exportar modelos ER de/para (alguns) gerenciadores de banco de dados (RDBMS) e editar (graficamente) o modelo físico de um banco de dados: tabelas, campos, e relacionamentos.

Ele também permite abrir e salvar modelos em formato XML, além de possuir suporte para operações de clipboard (copiar, recortar, colar), multiplos documentos (MDI), zoom, e comandos desfazer/refazer.

1.4 Motivação

Há diversos motivos possíveis para o desenvolvimento e o uso de um software como este. Entre eles:

- Os editores de ER mais usados são os próprios editores que vêm com o RDBMS em questão, como por exemplo o editor de tabelas/relacionamentos do Microsoft Access ou o Enterprise Manager do SQL Server, e desta forma não tem compatibilidade (para importar/exportar) com outros bancos. Eles nem sequer costumam permitir salvar o diagrama externamente ao gerenciador de dados. Também não costumam ter um bom suporte a documentação do sistema.

- Os melhores editores ER do mercado (ou seja, aqueles que permitem trabalhar com diversos gerenciadores, e salvar diagramas fora do banco) são proprietários e muito caros.

- A maioria dos editores gratuitos não suporta o Microsoft Access, que é um RDBMS cuja DDL (data definition language) ainda hoje foge um pouco da sintaxe ANSI SQL-92

- Apesar de haver RDBMS como o Oracle que possui versões para diversas plataformas, a maioria dos editores ER (mesmo os comerciais) roda em apenas uma plataforma (quase sempre Windows). Isto obriga o Analista (ou DBA) a fazer a modelagem sempre na mesma plataforma.

- A maioria dos editores tem interface confusa. Isto acaba escondendo do usuário (as vezes intencionalmente, tentando facilitar) o real funcionamento do modelo relacional

1.5 Minha experiência profissional com editores ER

Ao longo de minha experiência profissional percebi que os mais experientes analistas de sistemas (ou administradores de banco de dados, ou mesmo administradores de dados) costumam usar os editores bons, como o ERWin da Computer Associates.

Normalmente quem usa os editores dos próprios gerenciadores são principiantes (com excessão, é claro, para sistemas com poucas tabelas e pouco volume de dados, onde normalmente não é necessária uma boa modelagem), e este tipo de modelagem pode no futuro causar problemas de documentação, e migração, que se tivessem feito uso de um bom modelo de dados não aconteceria.

2 Introdução aos editores de ER

2.1 Principais Editores Comerciais

Sempre que eu falar em editor comercial estarei me referindo aos editores ER comerciais (não-gratuitos) que não acompanham nenhum gerenciador de banco de dados em especial, de modo que servem para diversos gerenciadores.

Os principais editores comerciais são estes:

- Computer Associates ERWin - $3,995.00 (sim, o nome não é coincidencia)
- Microsoft Visio - $499
- Sybase Powerdesigner - $2.500

Estas ferramentas CASE (computer aided software engineering) costumam ser altamente competitivas entre si: Todas as três já passaram por diversos donos, pois os fabricantes maiores acabam comprando os softwares de fabricantes menores.

2.2 Recursos mais usados

2.2.1 Índices

Além da edição de tabelas, atributos, chaves primárias, e relacionamentos (conceitos que estão fora do escopo deste documento), também costuma-se usar muito o recurso de índices. Índices são usados para acelerar a recuperação de dados, e basicamente são apontadores previamente ordenados para os registros de uma tabela. Assim como uma chave primária, também são um subconjunto das colunas da tabela.

Os editores costumam permitir 4 tipos de índices:

- Índice de PK - é um índice para as colunas da chave primária (PK) da tabela. Só pode haver um por tabela, é um índice único (não permite duplicados), obriga que todos seus campos sejam não-nulos (pois fazem parte da PK), e normalmente é o índice clustered da tabela (só pode haver um índice clustered por tabela, pois um índice clustered implica que os registros da tabela tenham a mesma ordenação que a do índice). É gerado automaticamente ao definirmos a chave primária.
- Índice de FK - é um índice para as colunas de uma chave estrangeira (FK) da tabela. Para cada conjunto de atributos que migraram para a tabela através de um relacionamento (ou seja, atributos que são chave primária em outra tabela), os editores criam um índice de FK, que agiliza as operações de JOIN entre as tabelas.
- Índice único - é um índice que assim como a chave primária não permite que haja um conjunto de valores duplicados nas colunas do índice. É também chamado de alternate key (AK), pois também identifica unicamente um registro na tabela, assim como a chave primária. Como os índices únicos e chaves primárias são as únicas chaves candidatas, todo relacionamento (atributos migrados) deve ser vinculado a um índice único ou índice de PK.

- Índice não-único - também chamado de inversion entry (IE), é simplesmente um índice em um conjunto de colunas, que permite duplicados, e serve apenas para acelerar a recuperação de dados.

2.2.2 Notação

Há diversas notações para modelos ER. As mais famosas são: Chen (ensinada na maioria dos cursos teóricos), IE (information engineering, ou crow's foot - pé de galinha), IDEF1X. No entanto, normalmente usa-se a notação IDEF1X ou IE. Optei pela IE, pois é aquela com a qual mais tive contato no trabalho. Além disso as notações são todas equivalentes e muito semelhantes entre si, de modo que quem conhece uma, não tem dificuldades em se adaptar a outra.

2.2.3 Subsistemas

A maioria dos modelos ER grandes costuma ser dividida em subsistemas - normalmente usando uma cor diferenciada para cada subsistema (para suas respectivas entidades). Optei por atender este requisito e permitir que cada entidade tenha uma cor diferente, permitindo a divisão do modelo em subsistemas.

2.2.4 Modelo Lógico e Físico

A grande maioria dos editores comerciais permite editar o modelo lógico e o modelo físico simultaneamente. Modelo lógico é aquele que considera as informações do negócio (entidades de dados) e seus relacionamentos derivados. Modelo físico é um mapeamento do modelo lógico que considera peculiaridades (como tipo de dados, indices, etc) do gerenciador de dados em questão.

Tenho observado (ao longo de minha experiência e também ao longo de minha pesquisa) que nem mesmo os mais experientes analistas costumam usar de fato o modelo lógico, normalmente partindo diretamente para a implementação física. Talvez por falta de conhecimentos teóricos, talvez por praticidade, mas o fato é que não vi ninguém que fizesse uso de um modelo lógico muito diferente do modelo físico. Desta forma, para facilitar o desenvolvimento (e por falta de um maior conhecimento nesta questão de lógico vs físico), optei por fazer apenas um editor de modelos físicos.

2.2.5 Outros

Outros recursos que também constatei que não são muito utilizados no modelo ER e portanto decidi não incorporar nesta primeira etapa são Domínios de dados, Filesystems, Defaults, Validations, e Triggers.

2.3 Recursos desejados

O ERWin possui uma interface muito interessante, pois apesar de assustar o usuário a principio, após ele se acostumar ele acaba gostando pois o software

mostra exatamente como funciona um modelo de dados.
O que quero dizer é que há alguns editores ER (como o do Microsoft Access e até o Enterprise Manager do Microsoft SQL) que acabam ”enganando” o usuário - muitas vezes dão (para citar um só exemplo) a impressão de que os relacionamentos entre entidades são feitos a nivel de coluna (ao usá-los, temos a sensação de que podemos associar qualquer coluna com qualquer coluna), quando na realidade um relacionamento - isto é, uma chave estrangeira - deve estar mapeado para uma chave candidata. A chave candidata normalmente é a própria chave primária da tabela (que é composta de uma ou mais colunas), mas, muita gente não sabe, pode também ser um índice único da tabela (também composto de uma ou mais colunas).

Editores como o do Access permitem o usuário ligar qualquer coluna com qualquer coluna (sem garantir integridade referencial) e outros como o do SQL Server fazem as devidas verificações nos mapeamentos (e criam a integridade referencial) mas não facilitam o trabalho do modelador disponibilizando a ele as chaves candidatas, de modo que se o usuário sair mapeando uma chave estrangeira pra colunas que não formem uma chave candidata (índice único ou PK), o gerenciador dá a mensagem de erro, sem maiores explicações sobre o que o usuário deve fazer.

Já no ERWin, o usuário simplesmente define quem é a entidade pai, a entidade filho, e os atributos da chave primária do pai (que identificam unicamente os registros) são migrados para a entidade filho. Também pode-se fazer o relacionamento apontando para qualquer índice único - o software já indica as chaves candidatas, que são os únicos possíveis atributos que identificam registros e desta forma podem ser migrados para uma entidade filho.

Já o Visio possui a estranha interface onde você primeiro arrasta tabelas da barra de ferramentas para o documento (até aí tudo bem) para depois arrastar os relacionamentos e vincular estas ”linhas soltas” com as respectivas tabelas, dando a (falsa) impressão de que o relacionamento é uma entidade que existe por si só, sem ter a necessidade de estar ligada a uma tabela. Apesar desta interface um tanto quanto esquisita (provavelmente derivada do fato do software fazer dezenas de tipos de diagramas diferentes, o que deve ter obrigado os modelos ER a adaptarem-se a interface de arrastar/soltar), o Visio possuia o interessante recurso de ”desfazer/refazer”, que decidi incorporar no meu aplicativo.

Desta forma optei por fazer uma interface semelhante a do ERWin (que possui uma UI muito mais lógica para um editor ER, apesar de não parecer tão amigável), mas incorporando alguns recursos do Visio.

Recursos desejados:

- Criar entidades (tabelas), mover, redimensionar
- Editar atributos (colunas)

- Editar indices e PKs (conjuntos de colunas)
- Editar relacionamentos (FKs)
- Multiplos documentos abertos simultaneamente
- Copiar/Colar (entre documentos ou no mesmo documento)
- Salvar/abrir modelos
- Importar/Exportar modelos
- Scrollbars
- Zoom
- Ver/editar propriedades do documento e de seus elementos (por exemplo, cores de background/foreground)

3 Ferramentas e técnicas utilizadas no desenvolvimento

3.1 Sobre o Microsoft .NET Framework

O Framework .NET da Microsoft é relativamente novo (o primeiro Beta surgiu em Julho de 2000). Não é apenas uma linguagem de programação, mas sim uma completa estrutura de desenvolvimento - um conjunto de bibliotecas (desde bibliotecas de graficos, estruturas de dados, bancos de dados, até Web Services, SOAP e XML), e uma interface de programação para acessar os serviços e APIs do Windows.

C# é a principal linguagem (juntamente com o sucessor do Visual Basic 6 - o Visual Basic.NET), e é uma linguagem muito parecida com o Java (assim como este, também foi baseado no C++), de modo que o aprendizado desta nova linguagem não foi muito ”doloroso”, visto que eu conhecia um pouco de Java. Já as características do Framework em si (e não da linguagem) e do Win32, estes sim foram complicados conforme explicarei mais adiante.

3.2 Sobre o DotGNU

O projeto DotGNU é parte do projeto GNU, e se propõe a ser para webservices e C# aquilo que o GNU/Linux está se tornando para aplicativos de desktop e servidor: um provedor de soluções gratuitas de software.

O principal projeto (dentre 3) é o DotGNU Portable.NET, que trata-se de uma implementação do CLI (Common Language Infrastructure, mais conhecido como .NET) - basicamente trata-se de um compilador, e um runtime engine. O

Portable.NET implementou alguns controles da biblioteca WinForms (do Microsoft .NET), de modo a ”simular” os mesmos controles que vemos no Windows. De modo semelhante ao SWING do Java, todos os controles foram implementados completamente, sem fazer uso de widgets como GTK, QT, WINE, etc. Além do Winforms (que torna-se o mais interessante por ter reconstruído controles nativos e funções da API WIN32 que só existiam no Windows), o Portable.NET também implementou todas outras bibliotecas principais (XML, ADO.NET, GDI+ etc).

O Portable.NET funciona em diversas CPUs (x86, ppc, arm, parisc, s390, ia64, alpha, mips, sparc) e sistemas operacionais (GNU/Linux [on PCs, Sparc, iPAQ, Sharp Zaurus, PlayStation 2, Xbox,...], *BSD, Cygwin/Mingw32, Mac OS X, Solaris, AIX), de modo que cumpre seu objetivo de (palavras dos proprios autores:) prevenir que a Microsoft concretize sua colocação de que ”the era of ’open computing,’ the free exchange of digital information that has defined the personal computer industry, is ending”.

O projeto é, obviamente, patrocinado pela Free Software Foundation

Até certa fase do projeto o Portable.NET estava atendendo minhas necessidades, no entanto, infelizmente conforme o projeto foi avançando e novos recursos foram sendo incorporados, descobri que o Portable.NET não havia implementado completamente algumas características usadas no meu projeto. Uma opção seria buscar alternativas (controles e bibliotecas alternativas para as funcionalidades que o Portable.NET ainda não possuia), mas acabei optando por me dedicar por enquanto apenas na versão para Windows, para após o primeiro release portar para o DotGNU.
Todo o código está bem modular, de modo que será razoavelmente fácil este procedimento. Em especial, duas implementações que não pude usar no Portable.NET foram o controle PropertyGrid do Windows, e as funcionalizades do namespace SoapFormatter (usado para a serialização que salva diagramas em XML).
Cheguei a entrar em contato com os desenvolvedores do projeto, e obtive a resposta de que já estariam trabalhando nestes dois - resta agora esperar para saber se vai demorar.

Maiores informações: http://www.dotgnu.org/

3.3 IDEs: Compiladores, Descompiladores, Debuggers

A principal ferramenta de desenvolvimento foi o Microsoft Visual Studio .NET 2003. Esta IDE possui uma interface mais amigável para fazer uso do compilador (que é gratuito e vem com o Framework), debugger, e até um gerador de documentação (comentários feitos em XML diretamente sobre variáveis e métodos, que apareciam no intellisense e na documentação gerada) muito útil para não se perder num projeto deste porte.

Apesar de a maioria dos artigos (nos diversos tópicos de .NET estudados) possuirem códigos pequenos, também tive um certo interesse em estudar alguns softwares maiores, cujo código não estava disponível. Para isto, descobri uma ferramenta chamada *Reflector for .NET* (http://www.aisto.com/roeder/dotnet/), que permitia descompilação de códigos, e uma melhor visualização de classes de qualquer software produzido em .NET. Desta forma pude tomar contato com aquilo que costuma se chamar de "best practices", ou mesmo "design patterns", que até então não sabia do que se tratava.

4 Trabalhos correlatos estudados

Além dos softwares comerciais já citados, também busquei por trabalhos semelhantes (gratuitos e de código aberto) para estudar.

4.1 Conheça seu adversário: Mogwai

Durante primeiros meses de projeto eu não encontrei nenhum editor ER gratuito. Recentemente, em uma nova busca, acabei descobrindo o Mogwai
(http://mogwai.sourceforge.net/erdesigner/erdesigner.html).
A princípio a idéia era semelhante - um editor ER gratuito, e de código aberto, feito em Java. Porém, logo nos primeiros testes eu constatei que não era nada do que eu queria:

- Só permite editar um documento por vez
- Sem clipboard. Sem Undo/Redo.
- Só trabalha conectado com o BD
- Só MySQL, PostgreSQL, Oracle, e outros incomuns (como Pervasive SQL).
- Os tipos de dados só ficam disponiveis após conectar com o BD

Fiquei extremamente desapontado com a colocação de que "Mogwai trabalha com os mais comuns bancos de dados - MySQL e Oracle". Como pode um editor de ER trabalhar com Pervasive SQL e não trabalhar com Microsoft SQL?
Ficava claro então que o ERWon poderia se tornar melhor que o Mogwai, ou ao menos competitivo, já que se propunha a atender dois gerenciadores muito usados - o Access e o Microsoft SQL Server.

5 Sistema Desenvolvido

Antes de tudo, creio que caiba dizer que minha experiência anterior com .NET se resumia quase que completamente a Web (ASP.NET, ou Web Forms), de modo que aprender o funcionamento do Windows Forms (WinForms) foi um pouco difícil. Algumas coisas como arrastar e soltar componentes realmente não são difíceis, e sucedem-se de maneira quase que intuitiva, assim como programação de eventos e propriedades destes objetos (que é quase idêntica ao Java, que eu já conhecia um pouco), e assim o projeto começou. No entanto alguns conceitos como MDI, conceitos sobre o funcionamento de GDI+ e sobre manipulação de shapes no documento, e mesmo o conhecimento sobre controles como o PropertyGrid só foram surgindo com o avanço do projeto.

Minha experiência com recursos e API do Win32 também era quase nula, visto que na maioria dos aplicativos para Windows que eu já havia feito, não havia usado recursos como GDI+, Clipboard e tratamento de eventos de Mouse.
Também não sabia como implementar um processador de comandos que me disponibilizasse comandos de desfazer e refazer as últimas operações executadas.
Por fim, apesar de já conhecer XML, tampouco foi útil, pois não conhecia o processo de serialização de dados - nem o conceito nem a forma como faria sua implementação em .NET.

Desta forma, o intuito inicial do projeto (que era aprender tecnologias novas) acabou se concretizando mais do que eu esperava, tendo tornado-se um projeto onde 80% (ou mais) dos desafios (e do tempo gasto) foram em questões de interface (UI), .NET e Win32 - a parte de bancos de dados em si acabou ficando para o final do projeto (ainda em desenvolvimento), visto que só poderiam ser concretizadas após eu ter uma interface funcional, e visto que eu já tinha certos conhecimentos de ADO.NET e definitivamente o acesso a dados não seria o mais complicado.

5.1 WinFoms - A Janela principal

Basicamente, chama-se de WinForms (ou Windows Forms) a todos os Formulários de aplicativos para desktop em .NET. No caso do ERWon, a janela principal trata-se de um WinForm com uma barra de menus (com menus para interagir com documentos e bancos de dados), uma barra de ferramentas de botões operacionais (o botão apertado define a operação atual), e um espaço vazio à direita onde serão exibidos e editados os documentos.

5.2 MDI

A idéia de permitir a edição de vários documentos simultaneamente é algo que a princípio eu não sabia como implementar. Demorei certo tempo até descobrir o conceito de MDI (Multiple Document Interface).
MDI é uma API do Windows (e obviamente "empacotada" por funções do WinForms no .NET) que permite aos programadores criarem facilmente aplicativos com múltiplas janelas. É uma forma de mostrar um windows form aonde há uma janela pai e várias janelas filhas (cada uma com seus próprios controles, como scroll, armazenando um documento diferente).
Vários dos aplicativos que usamos no dia-a-dia (Word, Excel, etc) possuem este conceito e fazem uso da API.
Em oposição ao MDI, há o SDI (Single Document Interface), exemplificado pelo

Windows Explorer, Notepad, etc... aonde somente uma janela age como interface.

A partir do momento em que descobri que era isto que eu precisava, foi simples encontrar exemplos de MDI em .NET e implementar algo semelhante. Dentro do form principal, cada documento é aberto em seu próprio form, chamados de ”MDI Child Form”.

Basicamente o formulário pai mantém uma lista de referencias aos formulários filhos (que também mantém um apontador para este formulário pai), além de um indicador de qual é o filho ativo (documento ativo).

5.3 UserControl/Documentos

Dentro de cada MDI Child Form (que, como o parent form, também é um WinForm) há um UserControl (uma espécie de componente criado pelo usuário). É neste UserControl que é feito todo o controle de scroll e zoom (pois estes são específicos de cada documento), interação com o usuário (selecionar objetos, mover, redimensionar, etc). Também é o Usercontrol que hospeda o documento, que por sua vez possui uma lista de elementos (entidades e relacionamentos).

O UserControl implementa a interface IScrollableControl, que permite fazer barras de scroll de maneira simples. Como eu não sabia que este recurso vinha pronto para usar, acabei construindo o sistema de scrolls ”na unha”, usando controles como o VerticalScrollBar e HorizontalScrollBar. Só depois fui perceber que foi uma grande perda de tempo, o que exemplifica o quanto é importante conhecer a plataforma que estamos trabalhando para usarmos melhor seus recursos e não precisar reinventar a roda.

5.4 PropertyGrid

O WinForm principal possui um controle chamado PropertyGrid, que permite visualizar e alterar propriedades de objetos selecionados.
No caso, é o Usercontrol (do documento ativo) que possui uma lista de elementos selecionados, e é esta lista de elementos selecionados é passada para o PropertyGrid. A partir desta lista, o PropertyGrid opera via reflexão (funções para obter a lista de propriedades e métodos de um objeto qualquer em tempo de execução) para descobrir quais são as propriedades públicas que os objetos selecionados tem em comum. Assim, se por exemplo, eu clicar em várias tabelas e no próprio documento como um todo, como todos eles (tabelas e documento) tem uma propriedade chamada BackGroundColor, a propriedade estará disponível no PropertyGrid e poderá ser vista/alterada para todos estes objetos simultaneamente.

O PropertyGrid é um controle que, infelizmente, ainda não foi portado para o DotGNU Portable.NET, de modo que atrapalhou o intuito inicial de ter um código idêntico rodando em Linux e Windows.

5.5 GDI+

GDI+ (Graphics Drawing Interface Plus) é uma API do Windows para programadores C/C++. Permite que os aplicativos usem gráficos e texto formatado tanto no vídeo quanto na impressora. Aplicativos baseados no GDI+ não acessam o hardware de gráfico diretamente, e sim através do GDI+ interagindo com os drivers de dispositivo. É o sistema de gráficos usado pelo Windows Forms.

O UserControl possui um evento de Paint, que por sua vez invoca para que cada entidade e relacionamento se desenhe sozinho. Este modelo descentralizado (cada elemento responsável por toda sua lógica) é um dos fundamentos da orientação a objetos (e design patterns) e permite uma melhor manutenção do código.
No caso, para evitar flickering, o Usercontrol pede que cada elemento se desenhe em um buffer criado especialmente, para evitar que o usuário veja os elementos sejam desenhados um a um. Após a conclusão do desenho deste buffer, aí sim o buffer aparece na tela, dando a impressão de que o Paint do Usercontrol foi instantâneo.

Tenho total consciência de que qualquer aluno que tenha cursado computação gráfica conheceria estes conceitos, mas como eu não havia cursado esta matéria, foi demorado para assimilá-los sozinho e colocá-los em prática.

5.6 Serialização

Serialização é o processo de transformar um (ou mais) objeto(s) em uma cadeia de bytes, que possa ser armazenada em um stream qualquer (memória, arquivo binário, xml, tcp/ip, etc). As classes para serem serializáveis devem ter um atributo [Serializable], que faz com que o formatador (o .NET possui serialização "nativa" através de dois formatadores, um binário e um XML) faça a serialização da classe automaticamente, que no caso salva todo o "grafo de dependências" do objeto.

Como usei serialização para as operações de clipboard (copy/paste), tive que obter um maior controle sobre a forma como os objetos eram serializados (visto que serializar "todo o grafo de dependências" não funcionaria a não ser para copiar um diagrama inteiro). Este maior controle foi possível através da implementação da interface ISerializable (e IDeserializationCallback), que nos permite um maior controle sobre a serialização podemos definir exatamente o que é serializado e como é serializado. Para isto há um método GetObjectData (que deve popular as propriedades que serão serializadas) e um construtor de serialização (que deve remontar as propriedades a partir do stream).

No projeto usei serialização para salvar diagramas em arquivo XML (que pode ser aberto em qualquer editor de texto, e tem um formato quase que ”legível”) e para implementar operações de clipboard.

Em uma fase inicial da UI, quando estava desenhando uma interface semelhante a do Visio, onde arrasta-se e solta-se os elementos no documento, também usei a serialização para fazer este Drag-Drop entre controles diferentes. Posteriormente este modelo de UI foi alterado (tentando assemelhar-se mais ao ERWin) e a serialização ficou sendo usada somente para XML e clipboard.

5.7 Clipboard

As operações de clipboard (recortar/copiar/colar) foram extremamente difíceis de serem implementadas, em grande parte devido a serialização de subconjuntos de elementos.

Basicamente, qualquer objeto que queremos copiar para o Clipboard (área de transferência) deve ser uma classe serializável e será armazenado em uma interface IDataObject, que permite armazenar (simultaneamente) objetos de diversos formatos.

Reconstruir os relacionamentos entre os objetos também envolveu um pouco de complicação, visto que a cópia de um objeto fazia referencias aos objetos originais - e não com suas cópias, para isso acabei extendendo algumas estruturas de dados. No caso, após a cópia de um subconjunto de elementos ainda era necessário verificar se as referências dos objetos copiados haviam sido todas copiadas ou se algumas referências deveriam ser quebradas (null), visto que a cópia não deveria ter nenhuma referência aos elementos originais.

Um recurso interessante do .NET que utilizei é a biblioteca Reflection, que permite obter informações sobre os objetos em tempo de execução. Há uma estrutura hierárquica de classes, e classes como entidade e relacionamento derivam de classes mais genéricas como Box e Line, que por sua vez derivam de uma classes genérica DocElement. Como a função para criar cópias de elementos (para operações de clipboard citadas acima) está na classe de DocElement (pois é usada por todas classes derivadas), a função usa reflexão para descobrir em tempo de execução o tipo (classe) dos objetos que está pedindo para ser copiado, e desta forma chama o construtor correto do objeto. Esta estrutura de elementos genéricos e elementos específicos permite que o software seja facilmente extendido no futuro.

5.8 Databinding em Winforms

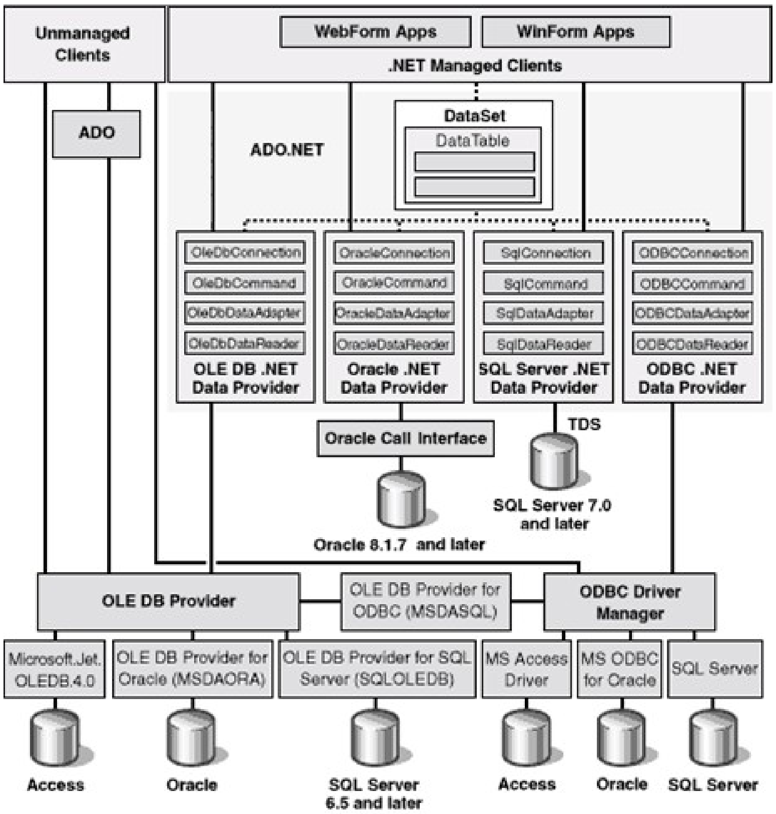
O formulário para editar os atributos das entidades é um Winform que é instanciado quando o usuário dá duplo-clique em uma entidade no documento. O

formulário permite a edição de colunas, índices e relacionamentos. Databinding trata-se de vincular controles (do Winforms) à dados, de modo que qualquer alteração nos controles da UI faz com que os dados sejam atualizados na memória, e vice-versa. Para isto, as coleções de Colunas (e classes derivadas, como por exemplo índices) devem implementar a interface IList, que permite que o Databinding funcione.
Devido a dificuldades com o databinding e a própria modelagem de colunas / índices / relacionamentos que levou certo tempo, este formulário ainda está em desenvolvimento - por enquanto está funcionando sem Databinding, monitorando diretamente os eventos dos controles.

5.9 ADO.NET

O ADO.NET é o sucessor do ADO (ActiveX Data Objects) - componentes para acesso a dados. A grande inovação em relação ao ADO tradicional é o fato de que o ADO.NET foi dividido em duas camadas - uma para trabalhar conectado com a fonte de dados, e uma para trabalhar desconectado.
No caso do ERWon, o funcionamento conectado ou desconectado é indiferente, visto que só usamos ADO.NET para fazer leitura das tabelas de sistema (e nunca para escrita).

Há certas diferenças na forma de conexão com os diversos gerenciadores. Para Oracle, temos o namespace (uma forma de organizar as classes e métodos, semelhante aos packages de java) System.Data.OracleClient; Para MS-SQL temos o System.Datta.SqlClient; Para qualquer banco acessível por OLE temos o System.Data.OleDb (o driver específico do banco é definido na string de conexão), etc. Para fazer uma conexão é necessário passar uma string de conexão, que varia para cada gerenciador e define informações como o nome do banco, usuário/senha, etc.



Este é um diagrama de como o ADO.NET se conecta (através de outros componentes) aos diversos gerenciadores de dados. Unmanaged clients se refere ao uso do ADO tradicional através de código não gerenciado (ex: C++, ou VB), enquanto .NET Managed Clients se refere ao uso do ADO.NET através de código gerenciado (.NET).

5.10 Engenharia Reversa

Engenharia reversa em um banco de dados consiste em transformar um schema em um diagrama Entidade-Relacionamento.
O processo é baseado na leitura e interpretação de metadata (dados que descrevem a própria estrutura dos dados do banco), que os bancos costumam disponibilizar na forma de tabelas de sistema.

Para este processo, nosso sistema se conecta (via ADO.NET e OLEDB.NET) aos gerenciadores de banco de dados, faz a leitura das tabelas de sistema, e interpreta as informações.
A informações têm a estrutura de um grafo acíclico dirigido, pois existe dependência (direcional) entre as tabelas.

Este recurso ainda não foi finalizado.

5.11 DDL e peculiaridades de cada RDBMS

Há diversas diferencas na geração de scripts DDL (para criar o banco de dados) para cada gerenciador.

Apesar das padronizações (ANSI SQL-92, SQL-99, etc), NENHUM gerenciador segue 100% das normas definidas. (Na verdade, há uma lista de cerca de 300 regras para um DBMS ser relacional, e nenhum produto comercial obedece todas elas). Desta forma está sendo um pouco complicado a geração dos scripts de criação de bancos, até porque os tipos de dados variam muito de gerenciador pra gerenciador.

Para citar um exemplo, o Microsoft Access (que na verdade é apenas um front-end para o JET Engine, responsável pelo gerenciamento de bancos em formato *.mdb) até recentemente não disponibilizava DDL ANSI SQL, de modo que dependendo da versão do banco ele deve ser manipulado via DAO (através de um modelo de objetos, e não de uma sintaxe ANSI). Além disso o Access não possui triggers, de modo que as opções de propagação são definidas diretamente nas propriedades do objeto Relation. Desta forma optei (forçosamente) a gerar bancos Access através da linguagem do DAO, assim como o ERWin faz.

Este recurso (geração de scripts DLL/DAO) ainda não foi implementado.

5.12 OOP Design Patterns

Como meu único contato com programação orientada a objetos foi em MAC-242 (Laboratório Digital 2), e um pouco em MAC-316 (Conceitos de Linguagens de Programação), foi um pouco inovador estar efetivamente programando com orientação a objetos. Acredito que um curso de MAC-413 (Tópicos de Programação Orientada a Objetos) ou mesmo MAC-441 (Programação Orientada a Objetos) teriam sido úteis para tornar o planejamento mais intuitivo.
Acabei me deparando com diversos ”estilos” e ”truques” nos códigos que estudei ao longo do projeto, e só depois fui descobrir tratarem-se de ”Design Patterns”, ou padrões de desenho de software OOP.
Não cheguei a efetivamente ler o famoso livro ”GOF” (E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995 - apelidado de ”Gang Of Four” book), mas cheguei a estudar alguns de seus principais padrões, ao longo dos artigos (e dos códigos) que estudei.

5.13 Algoritmos

Uma das implementações mais "elegantes" que tive ao longo do meu desenvolvimento foi a criação de comandos de "desfazer" e "refazer" (algo que nem o ERWin possuia). Demorei muito tempo pensando em como isso seria implementado, até que concluí que bastaria manter uma pilha onde armazenariamos os ultimos comandos, e apenas os dados necessários para desfazer o comando. A priori pode parecer óbvio, mas juro que nas primeiras vezes eu estava pensando em armazenar (para cada comando) o estado completo do diagrama, e não só as alterações feitas pelo comando.

Cada comando é uma classe derivada de uma classe abstrata que define os métodos Execute e Undo. Para ser usado ele é instanciado (através de um construtor que passa a ele os parametros da operação, que por exemplo no caso de uma movimentação na tela seriam a posição atual do mouse), e o método Execute é invocado. Após isso, se ele possuir um método Undo (nem todos comandos são desfazíveis) ele é armazenado na pilha de Undo (e se for invocado, vai para pilha do Redo). As pilhas de Undo e Redo são mantidas por um "Processador de Comandos", que cuida do gerenciamento automático das pilhas conforme os comandos são executados ou desfeitos.

Elegantemente simples. Mais elegante ainda pois usou uma estrutura de dados (pilha) que nem sempre usamos em nossos programas extra-academicos (pelo menos eu não costumava usar...), e pelo visto, as vezes não usamos simplesmente por não enxergar que trata-se estrutura mais apropriada.

Outra estrutura que surgiu deve-se a dependencia entre as tabelas. Para o diagrama ser processado (tanto para importação quando montamos o diagrama na memória, quanto para exportação quando geramos o script do banco), ele deve ser montado/gerado em uma ordem que não afete a dependência entre as tabelas. Cada restrição de FK (foreign-key) depende de uma outra tabela, deste modo devemos obter uma ordenação topológica que não viole a dependência entre as tabelas.

Por fim, conforme já explicado na seção de Clipboard, os algoritmos mais complicados foram aqueles que dão suporte às funcionalidades de serialização/clipboard. Para serializar subconjuntos de elementos, é necessário fazer uma cópia destes, restaurar os relacionamentos (referencias) originais (mas apontando para as cópias, e não para os originais), etc. Novamente, fiquei satisfeito por implementar uma estrutura "elegante" em um trabalho "não-academico".

5.14 Politica de tratamento de eventos da GUI

O Usercontrol é responsável pela captura de eventos de mouse - para mover tabelas, redimensionar tabelas (dependendo da posição da tabela que o usuário clica, trata-se de movimentação ou redimensionamento), selecionar tabelas, editar tabelas (nome/campos/indices/relacionamentos), etc.

Também é o responsável por monitorar o teclado - monitora teclas como ctrl e shift pois de acordo com seu estado (pressionadas ou não) os clicks do mouse devem agir de forma diferente (alterando a seleção ou adicionando objetos à seleção).

Foi necessário implementar certos truques, como a sobreposição (override) da função booleana que determina quais teclas são enviadas ao UserControl e quais teclas são tratadas pelo próprio .NET, implementado para que o controle pudesse capturar o pressionamento de teclas direcionais.

Novamente, creio que um curso de computação gráfica teria sido proveitoso. No entanto, fico feliz de (apesar de com mais dificuldade) ter vencido o desafio e conseguido implementar a interface gráfica em sua totalidade.

5.15 Tabelas de Sistema

A maioria dos gerenciadores disponibiliza metadados, que descrevem a própria estrutura dos dados do banco, na forma de tabelas de sistema. Através da interpretação destes metadados, é possível reconstruir um diagrama ER a partir do banco de dados.

Nas tabelas de sistema também há informações sobre alguns objetos do banco que o ERWon não implementa, como triggers e views. As tabelas de sistema que efetivamente são usadas na reconstrução do modelo ER são descritas abaixo para os gerenciadores estudados.

5.15.1 MS-SQL Server - system tables, ou system catalog

O Microsoft SQL Server possui as seguintes tabelas de sistema (chamado de catalogo de sistema):

- Sysobjects - Lista de todos objetos, incluindo tabelas.
- Sysindexes - Informações sobre índices.
- Sysindexkeys - Contém informações sobre colunas pertencentes aos índices.
- Syscolumns - Contém informações sobre colunas de tabelas (e de alguns outros objetos como visões).
- Sysconstraints - Mapeamento de constraints para os objetos donos.
- Sysforeignkeys - Contém informações sobre as chaves estrangeiras das tabelas.
- Sysreferences - Mapeia as chaves estrangeiras para as colunas a que elas se referem

- Sysdepends - Contém informações sobre dependência entre objetos.

5.15.2 Oracle - Data Dictionary

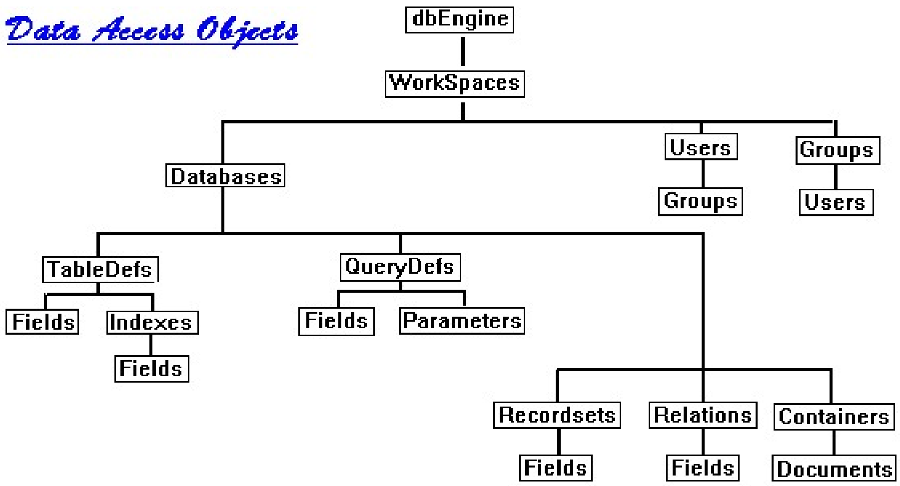
De modo muito semelhante, o Oracle possui as seguintes tabelas de sistema (chamado de dicionário de dados):

- USER_TABLES - Lista de tabelas
- USER_TAB_COLUMNS - Colunas de tabelas
- USER_CONSTRAINTS - Constraints (sejam de uma ou de múltiplas colunas), como chave primária, chave estrangeira, etc.
- USER_CONS_COLUMNS - Mapeia constraints para colunas. (uma constraint pode se referir a uma ou mais colunas)
- USER_INDEXES - Índices definido em colunas (tanto índices criados pelo usuário quanto índices gerados pelo próprio Oracle, que cria índices na PK se o usuário não criar, por exemplo).
- USER_IND_COLUMNS - Mapeamento de índices para colunas.
- USER_SEQUENCES - Lista de sequencias. No caso, os campos de AUTONUMERAÇÃO no Oracle são feitos através da criação de sequencias.

5.15.3 Microsoft Access

No caso do Access, em especial, além das tabelas de sistema (que devido a natureza do gerenciador não possuem todas as informações necessárias), há o modelo de objetos do DAO, que nos permite obter toda a estrutura do banco. Em particular temos as coleções:

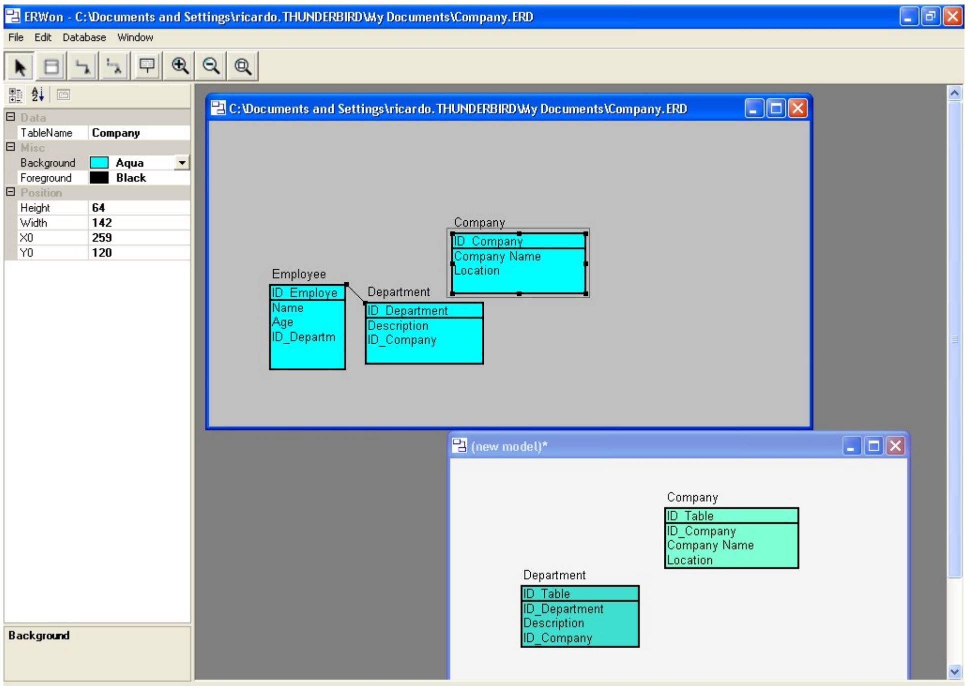
- Databases - coleção de bancos de dados abertos no atual Workspace do aplicativo Microsoft Access
- TableDefs - Tabelas de determinado banco de dados
- Fields - Colunas de determinada tabela
- Indexes - Índices de determinada tabela
- Fields - Colunas de determinado índice
- Relations - Relacionamentos (de chave estrangeira)

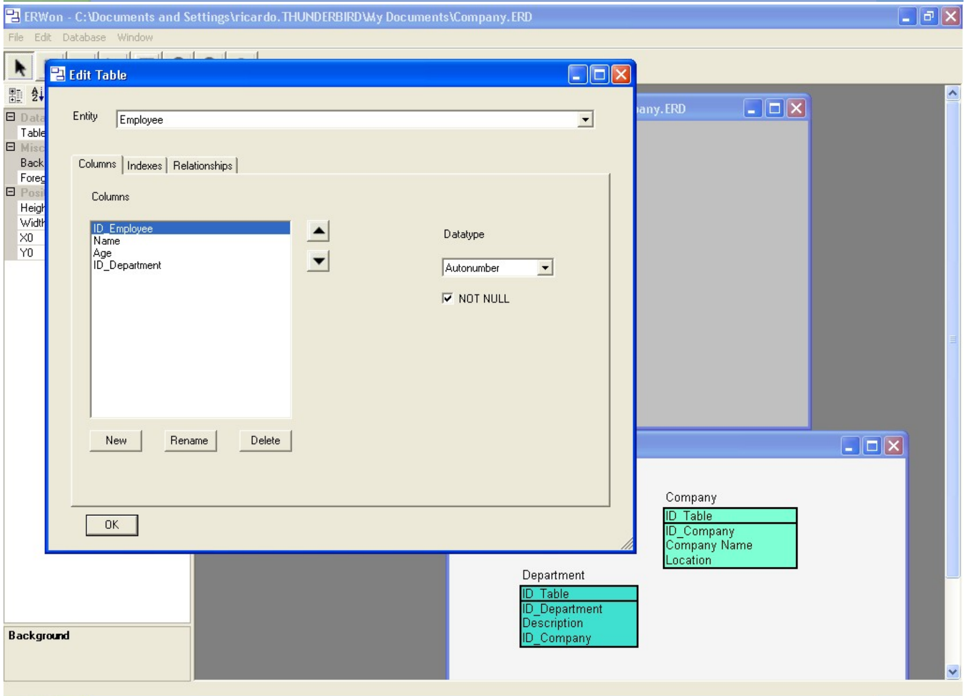


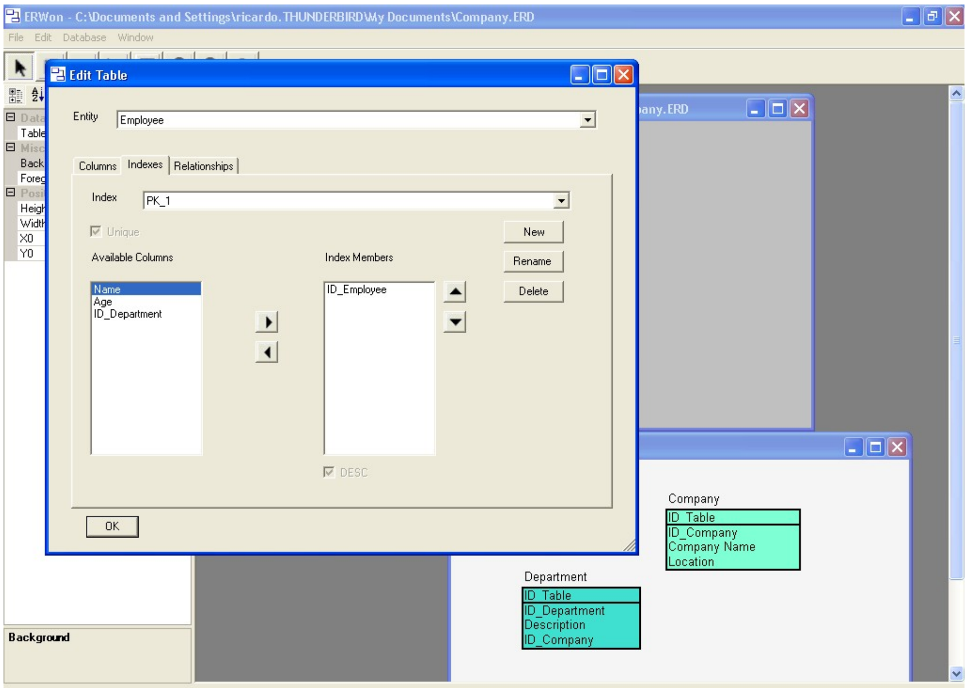
Este modelo de objetos pode ser explorado através do próprio Microsoft Access, que possui um editor de Macros (VBA) que permite navegar pelo modelo de objetos e examinar propriedades de qualquer banco mdb.

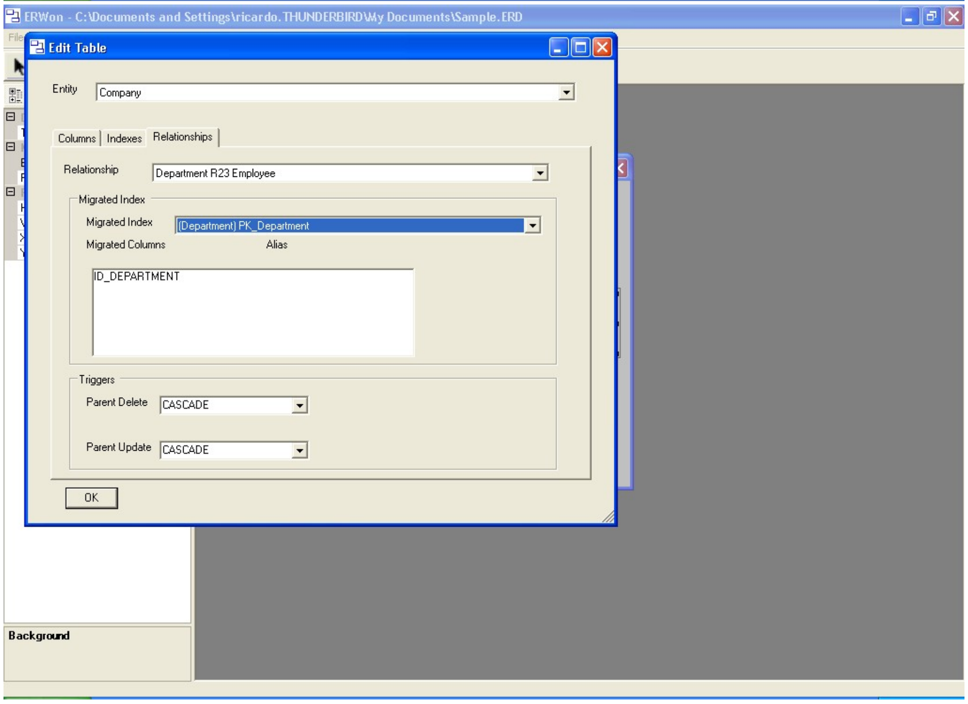
5.16 Resultado Final

Veja abaixo algumas telas do estado atual do programa.


ERWon - C:\Documents and Settings\ricardo.THUNDERBIRD\My Documents\Company.ERD
File Edit Database Window
Data
TableName Company
Misc
Background Aqua
Foreground Black
Position
Height 64
Width 142
X0 259
Y0 120
Background
C:\Documents and Settings\ricardo.THUNDERBIRD\My Documents\Company.ERD
Company
ID_Company
Company Name
Location
Employee
ID_Employe
Name
Age
ID_Departm
Department
ID_Department
Description
ID_Company
(new model)*
Company
ID_Table
ID_Company
Company Name
Location
Department
ID_Table
ID_Department
Description
ID_Company


ERWon - C:\Documents and Settings\ricardo.THUNDERBIRD\My Documents\Company.ERD
File Edit Database Window
Edit Table
Entity Employee
Columns Indexes Relationships
Columns
ID_Employee
Name
Age
ID_Department
Datatype
Autonumber
NOT NULL
New Rename Delete
OK
Background
Company
ID_Table
ID_Company
Company Name
Location
Department
ID_Table
ID_Department
Description
ID_Company


ERWon - C:\Documents and Settings\ricardo.THUNDERBIRD\My Documents\Company.ERD
Edit Table
Entity
Employee
Columns
Indexes
Relationships
Index
PK_1
Unique
New
Rename
Delete
Available Columns
Index Members
Name
Age
ID_Department
ID_Employee
DESC
OK
Company
ID_Table
ID_Company
Company Name
Location
Department
ID_Table
ID_Department
Description
ID_Company
Background


ERWon - C:\Documents and Settings\ricardo.THUNDERBIRD\My Documents\Sample.ERD
Edit Table
Entity
Company
Columns
Indexes
Relationships
Relationship
Department R23 Employee
Migrated Index
Migrated Index
(Department) PK_Department
Migrated Columns
Alias
ID_DEPARTMENT
Triggers
Parent Delete
CASCADE
Parent Update
CASCADE
OK
Background

6 Desafios, Frustrações e Recompensas

O trabalho que desenvolvi nos últimos meses foi bastante gratificante para mim, no sentido de que cumpriu o meu objetivo de adquirir mais bagagem prática em outras tecnologias além das ensinadas no BCC.
Durante o desenvolviento tive a oportunidade de estudar um pouco mais sobre a teoria relacional e sobre bancos de dados em si (que por mais que a gente conheça, sempre há muito mais a aprender), que é uma área que gosto muito, além também de estudar .NET, que é uma tecnologia com a qual tenho a cada dia me entusiasmado mais.

Como frustração, posso citar que tive alguns problemas pessoais que me obrigaram a iniciar um emprego em período integral, e desta forma acabei sendo obrigado a me dedicar ao projeto somente após o expediente, o que me impediu de ter uma maior interação com meu orientador, que tenho certeza seria muito proveitosa. Por outro lado, tive a feliz coincidência de que no emprego iniciado (Mondial Tecnologia em Informática) fui encaixado na divisão de .NET, de modo que as experiências e aprendizados do emprego costumavam servir para o projeto e vice-versa.

Como maior recompensa, tive a experiência de aprender na prática algumas das matérias que NÃO cursei no IME, como por exemplo Computação Gráfica, Laboratório de Banco de Dados, e Tópicos de Programação Orientada a Objetos. Foi ótima a sensação de saber que consegui aprender sozinho (guardadas as devidas diferenças entre o ensino acadêmico e os exemplos práticos que estudei).

Como já citado ao longo da monografia, e obviamente não podia deixar de recapitular nesta seção, um dos maiores desafios foi o de trabalhar com tecnologias que até então eu não tinha experiência. Trabalhar (aprender) conceitos como Clipboard, GDI+, Drag-and-Drop, Serialização/XML, MDI, WinForms, ADO.NET, PropertyGrid, etc, foram a maior dificuldade deste projeto, e me tomaram a maior parte do tempo. Devido a isso, acabei infelizmente não tendo tempo de terminar (até agora) a implementação da Importação/Exportação de modelos para os diversos gerenciadores, que só não foi a maior frustração devido ao fato de que acredito ter vencido barreiras muito maiores.

Um ponto importante, é que há uma EXTENSA documentação disponível na web para a maioria dos topicos citados, exceto para o GDI+ (computação gráfica, drag-and-drop de shapes, etc). Desta forma, uma das partes mais trabalhosas do projeto foram as rotinas para desenhar/mover/redimensionar tabelas. O desenho/movimentação de relacionamentos ainda está sendo implementado.

Outra frustração que tive (apesar de ainda não ter implementado a importação/exportação por inteiro) foi o fato de que não pude conseguir um servidor Oracle onde eu pudesse fazer meus testes, nem tampouco uma versão demo para testar.

7 Papel do curso e das disciplinas no desenvolvimento do projeto

Conforme já citado, disciplinas como Computação Gráfica, Topicos de POO, e Laboratório de Banco de Dados, teriam tornado o desenvolvimento do projeto um pouco mais rápido. No entanto, é inegável que a experiência nestes 4 anos de IME, fazendo trabalhos com conceitos que temos que pesquisar na internet e aprender sozinhos, aprendendo linguagens da noite para o dia, e acima de tudo adquirindo uma enorme capacidade de abstração, é inegável que esta experiência contribuiu muito para que eu pudesse sozinho vencer os desafios encontrados.

Posso afirmar que MAC-122 (Principios de Desenvolvimento de Algoritmos) e MAC-323 (Estruturas de Dados) são disciplinas que nos dão uma ótima base para programação, e nos ensinam muito sobre elegância e eficiência do código. Infelizmente, muitas vezes os alunos (especialmente aqueles que já trabalham e acabam se viciando nos estilos de programação usados por colegas de trabalho) não costumam empregar fora da faculdade. Tive a surpresa de poder usar diversas estruturas de dados e algoritmos elegantes neste trabalho, como nunca tinha usado antes em nenhum trabalho de minha vida profissional.

MAC-242 (Laboratório de Programação 2), apesar de não ter ensinado padrões de desenho OOP, foi a primeira disciplina que me ensinou uma linguagem orientada a objetos (Java), e desta forma foi importantissima para que eu pudesse entender como funcionam os padrões, e como usá-los para fazer uma boa modelagem no meu projeto.

MAC 328 - Algoritimos em Grafos - Esta disciplina foi aquela que mais me deu visão sobre algoritmos e estrutura de dados. O grafo é (na minha opinião) a estrutura ideal para praticar conceitos sobre estruturas de dados, e uma ótima estrutura para estudar algoritmos (corretude e eficiência), por isso é obrigatório citar como uma das matérias que considero mais importantes do curso.

MAC 426 - Sistemas de Bancos de Dados - apesar de ser uma matéria muito teórica e pouco prática, também ajudou um pouco.

8 Observações sobre relação entre o meio acadêmico (conceitos estudados) e o mercado (aplicações reais)

Creio que há um consenso geral entre os alunos que o curso do BCC é um curso voltado para a área acadêmica e não para o mercado de trabalho. A maioria dos alunos que se forma e parte para o primeiro emprego desconhece a maior parte das tecnologias que estão dominando o mercado, e creio que aquela história de que "as tecnologias mudam, os conceitos não" não é tão forte a ponto de permitirem que os alunos se formem sem ter uma base um pouco maior em tópicos "da moda", como orientação a objetos, banco de dados, sistemas distribuidos, web services, etc.

Creio que caibam algumas modificações ao currículo do curso - eu, pessoalmente, sugiro que as disciplinas Programação Orientada a Objetos, Tópicos de POO, e Laboratório de Banco de Dados sejam obrigatórias no currículo.

9 Que passos tomaria para aprimorar conhecimentos caso fosse continuar nesta atividade

Antes de tudo, espero até janeiro finalizar a primeira fase do projeto, onde falta apenas a finalização do formulário de edição de tabelas/relacionamentos (já em fase avançada) e a importação e exportação de modelos para os sistemas gerenciadores.

Caso eu continue com este projeto, o primeiro passo que faria seria um estudo aprofundado sobre padrões de desenho OOP. Em especial, fiquei com vontade de ler o livro do "GOF" (gang of four sobre design patterns, citado lá em cima no respectivo tópico). Tenho certeza que após ler um pouco (mais) sobre padrões, o código deve ficar muito mais elegante e reaproveitável.

Após ter adquirido um pouco mais de conhecimento em design patterns, suponho que o trabalho de extender o aplicativo se tornaria mais simples do que está sendo. Com uma melhor modelagem de classes eu poderia estar mapeando equivalências entre os tipos de dados dos diversos gerenciadores (assim como o ERWin permite, através do Datatype Standards Editor), e permitir assim que a migração de um gerenciador RDBMS para outro fosse mais automática e menos manual.

A título de curiosidade acadêmica, gostaria de estudar mais sobre diferenças entre modelos lógico e físico de um banco. Talvez, se eu descobrisse um bom motivo para os modelos serem diferentes (mesmo que a maioria dos usuários atualmente não use este tipo de recurso), pudesse fazer a implementação do modelo lógico, e quem sabe até fazer uma interface mais amigável aonde o usuário fosse de fato usar e entender para que servem os dois modelos.

Outra coisa importante a se pensar, seria que o gerador de script analisasse o banco atual e gerasse apenas scripts para atualizar o banco (alter table) de acordo com as diferenças que o modelo tem em relação ao banco.
Além disso, também seria interessante implementar recursos como triggers, defaults, validation, filesystems, etc. Para isso seria necessário um maior estudo sobre os diversos gerenciadores, pois a implementação destes recursos é bem diferente de gerenciador pra gerenciador.

Outra possibilidade para extender o aplicativo seria transformá-lo em um editor genérico de diagramas (como o Visio) ou mesmo em um editor de grafos, visto que as estruturas e idéias são bem semelhantes.

10 Bibliografia Utilizada

References

[1] MSDN Library - *http://msdn.microsoft.com/library/default.asp*

[2] Chris Sells - Windows Forms Programming in C# - Addison Wesley, 2004

[3] Nick Symmonds - *GDI+ Programming in C# and VB.NET* - Apress

[4] Joshi, Dickinson - Professional ADO.NET Programming - Wrox Press

[5] Thai, Lam - .NET Framework Essentials - O'Reilly, 2001

[6] Ferguson, Patterson, Beres, Boutquin, Gupta - C# Bible - Willey Publishing, 2002

[7] Julian Templeman, David Vitter - Visual Studio.NET - The .NET Framework Black Book - The Coriolis Group, 2002

[8] WinForms FAQ - *http://www.syncfusion.com/FAQ/WinForms/*

[9] Google - *http://www.google.com*

[10] The Code Project - *http://www.codeproject.com*

[11] DotNet247 - *http://www.dotnet247.com*

[12] Windows Forms.NET - *http://www.windowsforms.net*

[13] Patterns & Practices - *http://www.microsoft.com/resources/practices/default.mspx*

[14] Louis Davidson - *Professional SQL Server 2000 Database Design* - Wrox Press Ltd, 2001

[15] Robert Vieira - *Professional SQL Server 2000 Programming* - Wrox Press Ltd, 2001

[16] F.Scott Barker - Database Programming with Visual Basic.NET and ADO.NET - Sams Publishing, 2002

[17] Helen Feddema - *Expert One-on-One: Microsoft Access Application Development* - Wrox Press 2004

[18] Steven Roman - *Access Database Design & Programming* 2nd edition - O'Reilly, 1999

[19] Patricia Cardoza, Teresa Hennig, Graham Seach, Armen Stein - *Access 2003 VBA Programmer's Reference* - Wiley Publishing

[20] Professor Stumpf - *CIS 299 ERwin Tutorial*

[21] Rebecca M.Riordan - *Designing Relational Database Systems* 1999 - Microsoft Press