

WILSON XAVIER DA SILVA JUNIOR

**TRANSFERÊNCIA DE DADOS UTILIZANDO XML
FERRAMENTA EXCHANGE DATA XML - EDX**

MARINGÁ, MARÇO/2004

UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO: TECNOLOGIA EM
DESENVOLVIMENTO PARA WEB

Monografia apresentada ao departamento de Informática da UEM como parte dos requisitos para obtenção do título de Especialista em Tecnologia em Desenvolvimento para Web.

Orientador: Prof. Ademir Carniel

TRANSFERÊNCIA DE DADOS UTILIZANDO XML
FERRAMENTA EXCHANGE DATA XML - EDX

Wilson Xavier da Silva Junior

MARINGÁ, MARÇO/2004

WILSON XAVIER DA SILVA JUNIOR

**TRANSFERÊNCIA DE DADOS UTILIZANDO XML
FERRAMENTA EXCHANGE DATA XML - EDX**

Trabalho apresentado à Universidade Estadual de Maringá como requisito parcial para obtenção do título de Especialista em Tecnologia em Desenvolvimento para Web, sob a orientação do Professor Ademir Carniel.

Aprovado em: _____

BANCA EXAMINADORA

Prof. Ademir Carniel
(Universidade Estadual de Maringá)

Prof. Antonio Mendes da Silva Filho
(Universidade Estadual de Maringá)

Prof. Cesar Fernando Moro
(Universidade Estadual de Maringá)

DEDICATÓRIA

Dedico este trabalho a Deus, que nos momentos mais difíceis de nossas vidas nos abençoa com sua luz e nos guia para os caminhos da paz e da esperança.

AGRADECIMENTOS

Agradeço a meus pais, grandes incentivadores da minha jornada, que me deixaram como legado o caráter, a lealdade, a sinceridade e a perseverança.

Aos meus irmãos, que mesmo distantes sempre estiveram em meu coração em todos os passos da minha vida.

À minha noiva, companheira leal e dedicada que esteve sempre ao meu lado, me motivando e me confortando com seu carinho ao longo deste trabalho.

Aos meus amigos, que entenderam minhas ausências em prol do meu amadurecimento acadêmico e profissional.

E aos meus professores, fontes do conhecimento adquirido ao longo desta grande jornada.

LISTA DE TABELAS

Tabela 1: Registro de detalhe do padrão CNAB da FEBRABAN.	13
Tabela 2: Registro de detalhe do arquivo de retorno.	14
Tabela 3: Tabela de conversão de caracteres reservados.....	22

LISTA DE FIGURAS

Figura 1: Estrutura hierárquica de um intercâmbio.....	11
Figura 2: Estrutura de um arquivo de intercâmbio entre Bancos e Empresas.....	12
Figura 3: Código HTML exibido em um navegador.	20
Figura 4: Código XML exibido em um navegador.	21
Figura 5: Exemplo de um objeto COM.	28
Figura 6: Tela de configuração de uma conexão ODBC para Oracle.	31
Figura 7: Tela de configuração de uma conexão ODBC para MS-Access.	31
Figura 8: Componente TXMLDocument.	33
Figura 9: <i>Wizards</i> para criação de componentes <i>ActiveX</i>	35
Figura 10: <i>Wizard</i> para a criação de interfaces de um componente <i>ActiveX</i>	35
Figura 11: Exibição da estrutura de um componente <i>ActiveX</i> no Delphi.	36
Figura 12: Componente TADOConnection.	36
Figura 13: Tela de configuração do componente TADOConnection.	37
Figura 14: Tela de cadastro dos modelos de conexão.	40
Figura 15: Tela de cadastro das tabelas de conversão.	42
Figura 16: Tela de cadastro dos modelos de importação.	43
Figura 17: Tela de cadastro dos modelos de exportação.	43
Figura 18: Tela de cadastro dos modelos de migração.	44
Figura 19: Tela de execução dos processos de transferência de dados.	47
Figura 20: Diagrama de seqüência dos processos de cadastro dos modelos.	52
Figura 21: Diagrama de seqüência do processo de migração.	53
Figura 22: Diagrama de seqüência do processo de importação.	54
Figura 23: Diagrama de seqüência do processo de exportação.	55

SUMÁRIO

CAPÍTULO I – INTRODUÇÃO	1
1.1 Introdução	1
1.2 Motivação	2
1.3 Justificativa	4
1.4 Objetivos	5
1.4.1 Objetivos gerais	5
1.4.2 Objetivos específicos	5
1.5 Apresentação	6
CAPÍTULO II – FUNDAMENTAÇÃO TEÓRICA	7
2.1 Introdução	7
2.2 EDI (Electronic Data Interchange)	8
2.2.1 Definição e uso	8
2.2.2 Padrões	10
2.2.3 Internet e EDI	15
2.2.4 Futuro	17
2.3 Linguagens	18
2.4 DLL (Dynamic Link Library) ActiveX	27
2.5 Bancos de Dados e Acesso	30
2.6 Ferramenta de Programação	32
CAPÍTULO III – FERRAMENTA PROPOSTA	39
3.1 Apresentação	39
3.2 Funcionalidades	39
3.4 Implementação	48
3.4.1 EDX Interface	49
3.4.2 EDX Component	51
3.4.3 Modelagem	52
CAPÍTULO IV – CONCLUSÃO E TRABALHOS FUTUROS	57
4.1 Conclusão e Trabalhos Futuros	57
REFERÊNCIAS BIBLIOGRÁFICAS	60
APÊNDICES	62
Apêndice A – Diagrama de Classes	63
Apêndice B – Diagramas de Use-case	65
Efetuar cadastros	65

Executar transferências de dados.....	66
Apêndice C – Documentos DTD.....	67
CfgConexao.dtd	67
CfgTabelaConvesao.dtd.....	67
CfgMigracao.dtd	67
CfgExportacao.dtd	68
CfgImportacao.dtd.....	68
Intercambio.dtd	69

RESUMO

As empresas trocam informações eletronicamente, seja por obrigação, seja por necessidade. Como exemplos de imposição aparecem os bancos que, obrigatoriamente, trocam informações através de arquivos eletrônicos. Esta obrigação é estendida às empresas que realizam negócios com estes bancos, pois, as informações das transações (como envio e recebimento de títulos) são todas trocadas via arquivos texto com formatos específicos. Pelo lado da necessidade existem casos de empresas que fazem pedidos, compras ou trocam informações específicas via arquivos. Esta troca visa não só agilizar os processos como também reduzir os custos das transações. O presente trabalho vem propor uma forma de facilitar a transferência de dados, quer seja em uma migração (processo de efetuar a leitura em um determinado banco de dados e no mesmo processo atualizar um outro banco de dados), uma exportação (processo de leitura de um banco de dados e posterior geração de um arquivo contendo os dados pesquisados) quer seja uma importação (processo de leitura de um arquivo contendo dados em um formato específico e sua inclusão/atualização em um determinado banco de dados). Para tanto será proposta uma ferramenta que se baseia na linguagem XML, aproveitando desta linguagem os seus pontos fortes em relação ao intercâmbio de dados, a fim de criar modelos de transferência flexíveis e estruturados.

PALAVRAS-CHAVE: XML, Migração e Dados, Exportação de Dados, Importação de Dados

ABSTRACT

The companies change electronic information, either for obligation, either for necessity. As imposition examples appears the banks that, obligatorily, they change information through electronic files. This obligation is extended to the companies who carry through businesses with these banks, therefore, the information of transactions (as sending and act of receiving of headings) are all changed to way files text with specific formats. On the side of the necessity they exist companies that, purchases or change specific information by way of files. This change not only give speed to the processes as also to reduce the costs of the transactions. The present work comes to consider a form to facilitate the transference of data, either in a migration (process to effect the reading in one data base and in the same process to up date one another data base), an exportation (process of reading of a database and posterior generation of a file contending the researched data) either an importation (process of reading of an file contending data in a specific format and this inclusion/update in one determined data base). For in such a way a tool will be propose that is based on the XML language, using to advantage of this language its strong points in relation to the interchange of data, in order to create flexible and structuralized models of transference.

KEYWORDS: XML, Migration of Data, Exportation of Data, Importation of Data

CAPÍTULO I – INTRODUÇÃO

1.1 Introdução

O mercado globalizado, a crescente concorrência entre as empresas, clientes cada vez mais exigentes e conscientes de seu papel de reguladores dos preços e da qualidade dos produtos, faz com que as empresas busquem formas de reduzir cada vez mais seus custos, sem que isto afete a qualidade dos mesmos. Muitas vezes não há muito o que se fazer no processo produtivo para reduzir custos, ou talvez o investimento (tanto em equipamentos, quanto em material humano) para isto seja tão grande que a relação custo-benefício torne este processo inviável. Neste caso, uma redução de custos em processos como o da compra das matérias-primas e em outros materiais que indiretamente dão suporte ao processo (como materiais de expediente) pode auxiliar na redução dos custos da empresa, isto fatalmente se converterá em um balanço financeiro mais favorável.

Desde muito tempo, empresas trocam informações eletronicamente, seja por obrigação, seja por necessidade. Como exemplo de imposição tem-se os bancos que, obrigatoriamente, trocam informações através de arquivos eletrônicos. Esta obrigação é estendida às empresas que realizam negócios com estes bancos, pois, as informações das transações (como envio e recebimento de títulos) são todas trocadas via arquivos texto com formatos específicos.

Órgãos reguladores do governo, como a Agência Nacional de Saúde (ANS), exigem das empresas que forneçam informações eletronicamente como forma de monitorar o mercado e coibir, caso necessário, eventuais abusos e descumprimento das leis que regulamentam as atividades que eles fiscalizam.

Pelo lado da necessidade temos casos de empresas que fazem pedidos,

compras ou trocam informações específicas via arquivos. Esta troca visa não só agilizar os processos como também reduzir os custos das transações.

É importante descrever aqui a definição de alguns conceitos utilizados neste trabalho. Definiu-se como “Migração” o processo de efetuar a leitura em um determinado banco de dados e no mesmo processo atualizar um outro (ou até o mesmo) banco de dados. O termo “Exportação” caracteriza o processo de leitura de um banco de dados e posterior geração de um arquivo contendo os dados pesquisados. Para o termo “Importação” foi caracterizado o processo de leitura de um arquivo contendo dados em um formato específico e sua inclusão/atualização em um determinado banco de dados.

Ao longo deste trabalho serão apresentados ao leitor os motivos que nos levaram a desenvolver este projeto, e como este será capaz de facilitar o trabalho daqueles que necessitam realizar trocas de dados entre sistemas intra e extra-empresa.

1.2 Motivação

Nos últimos anos, as tecnologias de informação e comunicação vêm promovendo grandes mudanças no ambiente empresarial, principalmente na forma de organização do processo produtivo. Ao longo da cadeia produtiva essas tecnologias ampliam a comunicação entre as diversas empresas que realizam as atividades de transformação dos produtos, permitindo uma maior eficiência na coordenação da produção.

No relacionamento com os clientes, as empresas estão cada vez mais aprendendo a utilizar essas tecnologias para obter respostas mais precisas acerca da demanda dos produtos a serem fabricados, permitindo que as empresas planejem com maior precisão a escala do processo produtivo e conseqüentemente melhorando a coordenação. A comunicação mais eficaz com os clientes habilita que

estes passem a participar de forma mais eficaz na melhoria dos produtos e serviços, assim como permite o desenvolvimento de produtos e serviços que atendam as necessidades reais dos consumidores.

Apesar do comércio eletrônico já existir a pelo menos vinte anos, com a utilização de redes privadas do tipo EDI (*Electronic Data Interchange*), as transações comerciais realizadas nesse ambiente eram bastante reduzidas devido ao custo elevado das redes proprietárias, o que só era viável comercialmente para grandes empresas. Estas empresas necessitam dessa ferramenta para coordenação de suas atividades, principalmente, devido ao grande número de fornecedores ao longo da cadeia produtiva.

O surgimento da Internet como ambiente de transação comercial, uma rede aberta e global e de custos significativamente mais reduzidos que as redes proprietárias, permitiu a entrada de novos parceiros comerciais e clientes. Além disso, as características de interatividade, multimídia, processamento em rede e processamento de dados, habilitam o surgimento de diversas oportunidades para inovações organizacionais, comerciais, de *marketing* e no atendimento aos clientes.

Os processos internos da empresa também requerem uma atenção especial. A empresa deve buscar formas de aprimorar os processos que dão suporte a sua atividade fim. No caso de empresas que disponibilizam dados, produtos e serviços na Internet será necessário que os dados sejam publicados periodicamente (*on-line*, diariamente, semanalmente, mensalmente) para que estes não fiquem desatualizados.

O motivo que levou ao desenvolvimento deste trabalho foi o de criar mecanismos que auxiliem as empresas nos processos que dão suporte ao relacionamento desta com as outras empresas e com seus clientes.

1.3 Justificativa

Trocas de dados, em algumas situações, podem ser muito difíceis. O principal motivo para tal dificuldade é a grande diversidade de sistemas, de diferentes fornecedores, sendo que cada um deles tem sua própria estrutura de dados. A transferência dos dados de um sistema para outro, na maioria das vezes, requer que os dados de origem sejam transformados (mudança do tipo de dado ou alteração do dado em si) antes que sejam inseridos no sistema de destino.

Para cada tipo de transação (pedidos de compra, faturas, pedidos de cotação e notificações de remessa, por exemplo) pode ser necessária à elaboração de programas específicos para realizar a exportação e importação de tais informações. Levando-se em consideração que a estrutura de um banco de dados pode ser modificada ao longo do tempo, tais programas podem ser afetados criando assim uma necessidade de reprogramação dos mesmos ou até mesmo a necessidade de que sejam totalmente refeitos.

Uma ferramenta que permita, de forma flexível e simplificada, a criação de padrões pode facilitar os processos de intercâmbio de dados entre sistemas (dentro e fora da empresa).

Quer seja uma migração de dados entre sistema da própria empresa (inclui-se nisto as trocas de sistemas e publicação de dados para o banco acessado via Web), quer seja na transferência de dados entre empresas, são necessários, na maioria das vezes, programas específicos para cada tipo de transação (pedidos de compra, faturas, pedidos de cotação e notificações de remessa, por exemplo).

O ganho com uma ferramenta deste tipo poderá se dar tanto com a migração de dados dentro da própria empresa como no relacionamento entre empresas diferentes. Imagine fazer migrações de sistemas diferentes utilizando uma única ferramenta, sem que para isto seja necessário escrever uma única linha de código. Ou então enviar todos os seus pedidos a todos os seus fornecedores sem que para

isto se perca um tempo precioso de funcionários que poderiam estar realizando outras tarefas.

1.4 Objetivos

1.4.1 Objetivos gerais

Este trabalho tem como objetivo propor uma ferramenta que possa executar processos de migração de dados (fazer a leitura dos dados de um banco de dados de origem e inseri-los ou atualiza-los em um bando de destino) e realizar exportações e importações de dados via arquivos XML.

1.4.2 Objetivos específicos

- 1) Criar um componente responsável por executar os processos de migração de dados, criação dos documentos XML com os dados para serem importados por outros sistemas e a importação dos dados de um documento XML em um formatado específico suportado pela ferramenta. Este componente poderá ser executado tanto pela *interface* visual da ferramenta quanto por outro sistema que suporte a tecnologia a ser utilizada em seu desenvolvimento.
- 2) Elaborar documentos em XML para armazenar os dados necessários à realização dos processos de migração e transferência (exportação e importação) de dados.
- 3) Criar uma *interface* visual que permita o cadastramento dos modelos de transferência armazenados nos documentos XML.

1.5 Apresentação

No capítulo II serão descritos os fundamentos do EDI e apresentadas as tecnologias utilizadas na elaboração da ferramenta proposta.

O capítulo III será utilizado para apresentar os detalhes de implementação da ferramenta, com a descrição dos seus componentes e suas funções.

Finalizando o trabalho, o capítulo IV traz as considerações finais e trabalhos futuros.

CAPÍTULO II – FUNDAMENTAÇÃO TEÓRICA

2.1 Introdução

Para o desenvolvimento de um projeto de informática é necessário que além de definir o que será feito e como será o levantamento dos dados e processos, é imprescindível escolher quais serão as tecnologias e ferramentas empregadas em todo o processo de desenvolvimento. Como exemplos de itens a serem considerados têm-se: Sistemas Operacionais, Servidores de Rede, ferramentas Case, Sistemas Gerenciadores de Banco de Dados (SGDB's) e linguagens de programação. No mercado temos disponíveis várias implementações dos itens mencionados acima, cada uma tendo suas características, suas limitações e custos.

A escolha de tais itens podem tanto dificultar, atrasar e até mesmo tornar o projeto um fracasso quanto exceder as expectativas iniciais e levar o projeto ao sucesso. Foram analisadas diversas ferramentas de programação e tendências tecnológicas atuais, sendo que o objetivo foi encontrar as tecnologias que fossem mais adequadas ao projeto em questão. Como resultado disso foram selecionadas não só ferramentas de qualidade reconhecida, mas também tecnologias que atualmente têm despertado grande interesse da comunidade que trabalha com tecnologia da informação. Neste capítulo serão descritas as ferramentas e tecnologias empregadas no desenvolvimento do projeto, bem como o que motivou cada escolha.

Iniciando a discussão teórica deste capítulo temos as definições e um histórico do intercâmbio eletrônico de dados, um dos temas que serviram de base para o desenvolvimento deste projeto. O outro tema que também serviu de base para o projeto é a linguagem XML (*eXtensible Markup Language* – linguagem de marcação extensível) que será descrita nas tecnologias empregadas.

2.2 EDI (Electronic Data Interchange)

2.2.1 Definição e uso

Segundo O'Brien (2003), "o intercâmbio eletrônico de dados, EDI, envolve a troca de documentos de transação comercial por redes de computador entre parceiros comerciais (as organizações, seus clientes e fornecedores). Dados representando uma diversidade de documentos de transação comercial (tais como pedidos de compra, faturas, pedidos para cotações e notificações de remessa) são eletronicamente trocados entre computadores, utilizando formatos-padrão de mensagem para documentos".

Conforme Correia (1991), "70% das entradas de dados do computador de uma empresa, são originadas pelas saídas de dados de computadores de outras empresas sendo que 25% do custo das transações derivam da entrada de dados". Seguindo este raciocínio tem-se que os dados gerados por esta empresa fatalmente serão introduzidos nos sistemas de outras empresas, gerando uma grande cadeia de troca de informações. Além dos possíveis erros dos dados recebidos (neste caso, um algoritmo de crítica deve ser aplicado sobre os dados para minimizar as inconsistências) também podem ocorrer erros durante a digitação de tais entradas. Toda esta redundância administrativa acarreta perda de tempo e gastos que poderiam ser evitados caso as entradas fossem processadas com um mínimo de interferência humana.

O envolvimento de grandes empresas e bancos em tais transações comerciais eletrônicas têm se estendido por décadas, sendo necessário o suporte de serviços de outras empresas como administradoras de cartão de crédito e caixas eletrônicos. Tradicionalmente para tais serviços estas empresas tiveram que fazer uso das *Value Added Networks* (VANs) com a finalidade de compartilhar os dados necessários às suas transações comerciais eletrônicas (ANDERSON *et al*, 2001).

O desenvolvimento do EDI se deu devido à necessidade de comunicação eficiente entre parceiros comerciais. No mundo dos negócios, tradicionalmente, a comunicação pode acontecer de duas formas: não-estruturada (mensagens, memorando e cartas) e estruturada (pedidos de compras, avisos de despacho de mercadorias, faturas e pagamentos). O EDI abrange a troca de mensagens estruturadas enquanto as aplicações de Correio Eletrônico tratam das comunicações não-estruturadas. Em uma mensagem estruturada os dados são formatados com base em um padrão preestabelecido, de modo a facilitar a transferência eletrônica entre sistemas de computadores (COLCHER e VALLE, 2000).

Para Colcher e Valle (2000), no entanto, as vantagens da utilização do EDI ainda não podem ser generalizadas para todos os setores, visto que os segmentos mais favoráveis à utilização do EDI são aqueles onde as empresas possuem um relacionamento relativamente estável entre seus parceiros e onde o número de transações comerciais efetuadas é grande. Ainda para os autores, o principal mercado de EDI no Brasil está concentrado hoje, basicamente, em quatro segmentos: mercantil (com 28 % do mercado – onde estão as empresas de comércio varejista e atacadista e seus fornecedores); automotivo (que detém 8 % do mercado – compreendendo as empresas montadoras, seus fornecedores de componentes, peças, material de escritório e as concessionárias); financeiro (dono de 40 % do mercado – que inclui os bancos e seus clientes pessoa jurídica); e logística e transporte (responsável por 5 % do mercado – envolvendo empresas de transporte rodoviário, acondicionamento e distribuição de mercadorias e seus clientes). No restante do mercado, vale ressaltar o crescimento dos setores fármaco (cujo sistema conecta os laboratórios às redes de distribuição de medicamentos) e de seguros (onde sistemas de EDI interligam as entidades públicas e privadas).

No entanto, o crescimento do relacionamento entre empresas via EDI encontra obstáculos justamente naquilo que é a base para o seu funcionamento, a padronização. O fato de o EDI estar associado à adoção de um padrão predefinido, acaba impondo certa rigidez às transações comerciais praticadas pelas empresas. Devido a isso, as empresas que realmente pode ser beneficiadas com a utilização do EDI são aquelas que possuem um grande número de transações que possam ser

padronizadas, justificando a perda de flexibilidade ao adotar um padrão de intercâmbio já estabelecido.

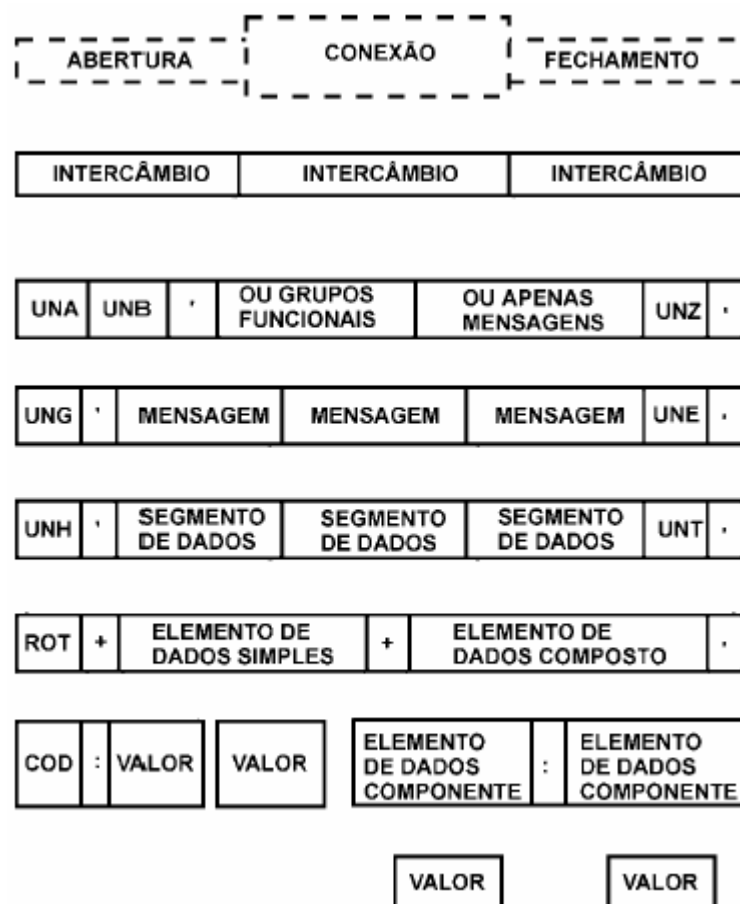
2.2.2 Padrões

No início da implementação do EDI foram desenvolvidos formatos para atender, isoladamente, às necessidades individuais de cada empresa. Rapidamente os usuários perceberam as limitações desses padrões proprietários. Desse modo novos padrões industriais foram desenvolvidos para atender às necessidades da comunidade empresarial. Entretanto, as companhias envolvidas em comércio com diversos setores industriais ainda enfrentavam barreiras e, conseqüentemente, a necessidade de padrões nacionais tornou-se clara (COLCHER e VALLE, 2000).

Atualmente existe uma grande diversidade de padrões para o EDI, sendo a grande maioria deles criados na fase inicial do aparecimento desta tecnologia. Os primeiros padrões refletiam as especificações das transações comerciais de cada setor da economia e do ambiente empresarial de cada país. Com o crescente aumento no número e na diversidade de usuários de EDI, surgiram problemas na troca de informações entre diferentes setores e países, limitando sua utilização. Em virtude disso foram sendo incentivadas iniciativas no intuito de se estabelecer padrões mais abrangentes através da negociação de acordos internacionais. Os comitês criados para este fim deram origem a padrões como o Odette (*Organisation for Data Exchange by Tele Transmission in Europe*) que foi adotado no setor automobilístico europeu, e o ANSI X12 criado pelo *American National Standard Institute*, que propunha um formato comum para as diversas transações comerciais envolvidas nas relações intra e inter industriais. No entanto, o grande avanço no sentido da harmonização dos padrões de mensagens veio quando a Organização das Nações Unidas (ONU) apresentou o padrão UN/EDIFACT (*United Nations/Electronic Data Interchange for Administration, Commerce and Transportation* – Nações Unidas/Intercâmbio Eletrônico de Dados para Administração, Comércio e Transporte) e este foi aceito em 1987 como padrão ISO

9735 (*International Standards Organization*). A importância do surgimento do padrão UN/EDIFACT (conhecido também por apenas EDIFACT) pode ser medida pelo fato dos comitês gestores dos padrões de EDI mais importantes – o próprio EDIFACT juntamente com o ANSI X12 – promoverem diversas iniciativas visando a convergência dos padrões tomando como base o EDIFACT (COLCHER e VALLE, 2000).

No padrão EDIFACT existem diversos grupos funcionais ou mensagens dentro de um intercâmbio e diversas mensagens em um grupo funcional. Uma mensagem, por sua vez, consiste de segmentos. A Figura 1 ilustra a hierarquia de um intercâmbio EDIFACT.



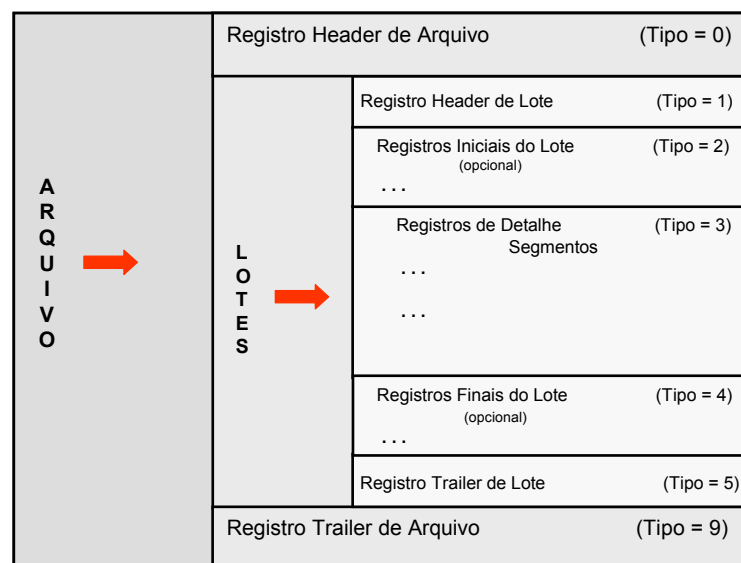
Fonte: Colcher e Valle, 2000.

Figura 1: Estrutura hierárquica de um intercâmbio.

No Brasil, representado o setor que detém a maior parte do mercado de EDI, a FEBRABAN (Federação Brasileira de Bancos), através do Centro Nacional para

Automação Bancária (CNAB), criou seu próprio padrão de troca de dados. Contudo, o padrão CNAB foi elaborado de modo a deixar em aberto alguns campos do formulário das mensagens para que os bancos tivessem espaço adicional para eventuais acréscimos de informações específicas que não foram previstas no modelo original. Esta abertura acabou por desencadear o aparecimento de várias versões do padrão CNAB, desenvolvidas e adotadas pelos sistemas de EDI de cada banco. Em virtude disso, o que se convencionou chamar de “padrão CNAB” é na verdade um conjunto de versões incompatíveis com o modelo original. A perspectiva é de que o “padrão CNAB” venha a ser substituído em pouco tempo por uma solução de EDI baseado no EDIFACT (COLCHER e VALLE, 2000).

Atualmente, o arquivo de troca de informações entre Bancos e Empresas é composto de um registro *header* de arquivo, um ou mais lotes de Serviço / Produto e um registro *trailer* de arquivo, conforme ilustra a Figura 3 abaixo:



Fonte: FEBRABAN, 2003.

Figura 2: Estrutura de um arquivo de intercâmbio entre Bancos e Empresas.

Conforme a estrutura apresentada na Figura 2, um único arquivo pode conter vários lotes de Serviços/Produtos distintos. Este procedimento, que permite às Empresas e Bancos consolidarem em um só arquivo todas as informações que desejam trocar entre si, deve ser previamente acordado entre cada Banco e Empresa Cliente (FEBRABAN, 2003).

As Tabelas 1 e 2 ilustram a diferença entre o padrão elaborado pela FEBRABAN e o que realmente é utilizado pelos bancos. Os registros representados pelas tabelas a seguir são referentes ao registro de detalhe de um arquivo de retorno (arquivo enviado pelo banco às empresas) com as informações de pagamento de títulos.

Campo			Posição		Nº Dig	Nº Dec	Formato	Default	Dês-crição			
			De	Até								
01.3J	Controle	Banco	Código no Banco da Compensação		1	3	3	-	Num		G001	
02.3J		Lote	Lote de Serviço		4	7	4	-	Num		*G002	
03.3J		Registro	Tipo de Registro		8	8	1	-	Num	'3'	*G003	
04.3J	Serviço	Nº do Registro		Nº Seqüencial do Registro no Lote		9	13	5	-	Num		*G038
05.3J		Segmento		Código de Segmento no Reg. Detalhe		14	14	1	-	Alfa	'J'	*G039
06.3J		Movimento	Tipo	Tipo de Movimento		15	15	1	-	Num		*G060
07.3J			Código	Código da Instrução p/ Movimento		16	17	2	-	Num		G061
08.3J	Pagamento	Código Barras		Código de Barras		18	61	44	-	Num		*G063
09.3J		Nome do Cedente		Nome do Cedente		62	91	30	-	Alfa		G013
10.3J		Data Vencimento		Data do Vencimento (Nominal)		92	99	8	-	Num		G044
11.3J		Valor do Título		Valor do Título (Nominal)		100	114	13	2	Num		G042
12.3J		Desconto		Valor do Desconto + Abatimento		115	129	13	2	Num		L002
13.3J		Acréscimos		Valor da Mora + Multa		130	144	13	2	Num		L003
14.3J		Data Pagamento		Data do Pagamento		145	152	8	-	Num		P009
15.3J		Valor Pagamento		Valor do Pagamento		153	167	13	2	Num		P010
16.3J		Quantidade da Moeda		Quantidade da Moeda		168	182	10	5	Num		G041
17.3J		Referência Sacado		Nº do Docto Atribuído pela Empresa		183	202	20	-	Alfa		G064
18.3J	Nosso Número		Nº do Docto Atribuído pelo Banco		203	222	20	-	Alfa		*G043	
19.3J	Código de Moeda		Código de Moeda		223	224	2	-	Num		*G065	
20.3J	CNAB		Uso Exclusivo FEBRABAN/CNAB		225	230	6	-	Alfa	Branco	G004	
21.3J	Ocorrências		Códigos das Ocorrências p/ Retorno		231	240	10	-	Alfa		*G059	

Fonte: FEBRABAN, 2003.

Tabela 1: Registro de detalhe do padrão CNAB da FEBRABAN.

O registro representado pela Tabela 1 é utilizado tanto no arquivo de envio (arquivo enviado pela empresa aos bancos) quanto no arquivo de retorno. A Tabela 2 apresentada a seguir é utilizada pelo banco Itaú e a formatação do registro representado por ela é utilizada apenas nos arquivos de retorno enviados pelas empresas ao banco. Vale destacar que o padrão elaborado pela FEBRABAN utiliza 240 caracteres para compor o registro, enquanto que o arquivo elaborado pelo Itaú (que é baseado no padrão CNAB) faz uso de 400 caracteres para definir as informações necessárias ao processamento do Banco.

Nome do Campo	Significado	Posição	Picture	Conteúdo
Código de Registro	Identificação do registro Transação	001 001	9(01)	1
Código de Inscrição	Identificação do tipo de inscrição/empresa	002 003	9(02)	01=CPF 02=CGC
Número de Inscrição	Número de inscrição da empresa (CPF/CGC)	004 017	9(14)	
Código da Empresa	Identificação da Empresa no Banco	018 029	9(12)	Nota 1
Branços	Complemento de registro	030 037	X(08)	
Uso da Empresa	Identificação do Título na empresa	038 062	X(25)	Nota 3
Nosso Número	Identificação do Título no Banco	063 070	9(08)	Nota 4
Branços	Complemento do registro	071 082	X(12)	
carteira	numero da carteira	083 085	9(03)	NOTA 6
nosso número	Identificação do Título no Banco	086 093	9(08)	
DAC nosso número	dac DO NOSSO NÚMERO	094 094	9(01)	
Branços	COMPLEMENTO do registro	095 107	X(13)	
Carteira	Código da Carteira	108 108	X(01)	Nota 6
Cód. de Ocorrência	Identificação da Ocorrência	109 110	9(02)	Nota 20
Data de Ocorrência	Data de ocorrência no Banco	111 116	x(06)	DDMMAA
Seu Número	Nº do documento de cobrança (Dupl, NP etc)	117 126	x(10)	NOTA 21
Nosso Número	Confirmação do número do título no banco	127 134	9(08)	
Branços	Complemento do registro	135 146	X(12)	
Vencimento	Data de vencimento do título	147 152	9(06)	DDMMAA
Valor do Título	Valor Nominal do Título	153 165	9(11)V9(2)	
Código do Banco	Número do Banco na câmara de compensação	166 168	9(03)	
Agência Cobradora	Ag. cobradora, ag. de liquidação ou baixa	169 172	9(04)	Nota 10
Dac Ag. Cobradora	Dac da agência cobradora	173 173	9(01)	
Espécie	Espécie do Título	174 175	9(02)	Nota 11
Tarifa de Cobrança	Valor da despesa de cobrança	176 188	9(11)V9(2)	
Branços	Complemento do registro	189 214	X(26)	
Valor do IOF	Valor do IOF a ser recolhido (Notas Seguro)	215 227	9(11)V9(2)	
Valor Abatimento	Valor do abatimento concedido	228 240	9(11)V((2)	NOTA 22
Descontos	Valor do desconto concedido	241 253	9(11)V9(2)	NOTA 22
Valor Principal	Valor lançado em conta corrente	254 266	9(11)V9(2)	
Juros de mora/multa	Valor de mora e multa pagos pelo sacado	267 279	9(11)V9(2)	
BRANCOS	COMPLEMENTO DO REGISTRO	280 295	X(16)	
DATA CRÉDITO	DATA DE CRÉDITO DESTA LIQUIDAÇÃO	296 301	x(06)	DDMMAA
instr.cancelada	código da instrução cancelada	302 305	9(04)	nota 23
brancos	complemento de registro	306 324	x(19)	
Nome do Sacado	Nome do Sacado	325 354	X(30)	
Branços	COMPLEMENTO do registro	355 377	X(23)	
Erros	registros rejeitados ou alegação do sacado	378 385	X(08)	Nota 23
Branços	Complemento do registro	386 392	X(09)	
CÓD. DE LIQUIDAÇÃO	MEIO PELO QUAL O TÍTULO FOI LIQUIDADO	393 394	X(02)	NOTA 32
Número Seqüencial	Número seqüencial do registro no arquivo	395 400	9(06)	
X = Alfanumérico	9 = Numérico	V = Vírgula Decimal Assumida		

Fonte: Itaú, 2000.

Tabela 2: Registro de detalhe do arquivo de retorno.

2.2.3 Internet e EDI

Quando da criação da Internet, seu objetivo inicial foi o de trocar e exibir documentos estáticos legíveis pelo ser humano, contudo, acabou se tornando um canal essencial de comunicação, propaganda e vendas para organizações de todos os tipos ao redor do planeta. Ao longo dos anos, a Internet sofreu uma rápida e grande evolução, passando da troca de documentos científicos a um meio onde se pode comprar ou vender uma infinidade de bens e serviços. Estas funções de compra e venda são tipicamente qualificadas como itens do comércio eletrônico, mais conhecido como *e-commerce*. Nas palavras de Anderson (2001), “*e-commerce* é a troca de bens ou serviços por dinheiro, entre duas ou mais entidades através da Web”.

Apenas citando dois casos de empresas que fazem negócio pela Internet temos as Lojas Americanas e a companhia aérea GOL. No caso das Lojas Americanas a empresa disponibilizou um portal (que pode ser conferido em <http://www.americancas.com>, visitado em janeiro de 2004) onde os clientes podem efetuar suas compras no conforto de sua casa, podendo escolher as formas de pagamento e receber os produtos adquiridos sem o incômodo de se dirigirem à loja (isto se existir uma loja da empresa em sua cidade). Já no caso da GOL (companhia de transporte aéreo de passageiros), a empresa disponibiliza passagens para qualquer um dos trechos da companhia em seu site (visitado em janeiro de 2004 no endereço <http://www.voegol.com.br>), sem que o cliente tenha que enfrentar as filas dos aeroportos. Outro grande diferencial deste caso da GOL é que as passagens compradas pelo site da companhia são mais baratos que os comprados direto no balcão ou pelo televendas da empresa.

Pode-se constatar então que a Internet passou de um canal de propagação e troca de informação para um poderoso canal de distribuição de bens e serviços, mudando de forma significativa a maneira de se fazer negócios atualmente.

Para um usuário da Internet um site de comércio eletrônico tem apenas a

visão final do processo, onde o produto é escolhido e entregue ao cliente no endereço que este escolher. Entretanto, o processo realizado por trás desta simples visão do usuário final vem sendo automatizado utilizando o EDI ou outros formatos e protocolos patenteados. Além disso, existem grandes áreas de negócios que utilizam o *e-commerce* para realizar negócios com outras empresas. A este relacionamento comercial entre empresas através da internet dá-se o nome de *business-to-business* (ANDERSON *et al*, 2001).

Nos últimos anos, muito tem se falado sobre a Internet, sobretudo nos benefícios do comércio eletrônico. Esta crescente discussão e propaganda em torno da Internet poderiam levar uma pessoa a acreditar que conceitos como comércio eletrônico não existiam antes de a Internet tornar-se uma palavra de uso doméstico (MACGRATH, 1999).

No entanto, Anderson (2001) aponta que o EDI existe há uns 30 anos, sendo que este teve um papel significativo na preparação do caminho para o *e-business* (denominação para negócios feitos pela Internet). Muito antes da Internet as empresas trocavam dados eletronicamente utilizando transações padronizadas preparadas pra cada ramo da indústria. O autor cita ainda que vários setores da indústria como o automotivo, de alimentos e eletrônicos, obtiveram uma economia significativa e melhora nos processos corporativos usando o EDI.

O'Brien afirma que "utilizar EDI é uma das maneiras pela qual uma empresa pode formar laços comerciais estratégicos com seus clientes, fornecedores e outros parceiros comerciais". Nesta linha de raciocínio não devemos colocar apenas as empresas que focam, total ou parcialmente, seus negócios na Internet. A troca eletrônica de dados vem como complemento de uma estruturação visando o amadurecimento das relações comerciais entre as empresas.

A troca eletrônica de dados para empresas com seu ramo de atividade voltado para a Internet torna-se um procedimento não apenas natural, mas também obrigatório. Como dito anteriormente, uma empresa deve buscar agilizar seus processos; transportando isso para a Internet essa necessidade se torna

imprescindível para a sobrevivência da empresa no mercado.

2.2.4 Futuro

Anderson (2001) relata que os modelos de EDI estão sendo revisados para que este seja mesclado com uma nova tecnologia que promete reduzir os custos e facilitar o acesso à tecnologia por qualquer empresa. Esta tecnologia vem a ser a linguagem XML.

Já Ogbuji (1999) descreve que muitos na comunidade do intercâmbio eletrônico de dados (EDI) encontraram na XML uma forma de aumentar a adoção do EDI, que tradicionalmente sempre foi caro e complexo de ser implementado. Muitas empresas pequenas ou médias estão sofrendo pressão para suportar o EDI e esta pressão vem geralmente dos fornecedores, dos clientes, ou dos distribuidores maiores que investiram muito em EDI a fim cortar seus custos de negociação. No entanto, o autor ainda afirma que o sucesso do EDI baseado em XML depende dos padrões sólidos para XML e das tecnologias relacionadas, do apoio contínuo de vendedores de *software* e o apoio de desenvolvedores de EDI tradicional. Todos estes fatores estão recebendo bastante atenção e, uma vez que esta tecnologia esteja dominada, as empresas que se utilizarem dela poderão obter muitos benefícios em suas transações comerciais, melhorando seu relacionamento entre seus fornecedores e clientes e diminuindo os custos do negócio. Esta proposta de se fazer EDI baseado em XML é denominada XML/EDI.

As características da linguagem XML, sua estrutura e algumas de suas aplicações serão discutidas no próximo tópico deste capítulo.

2.3 Linguagens

No início, quando a HTML (*Hiper Text Markup Language* – linguagem de marcação de hipertexto) foi desenvolvida, tinha-se como preocupação apenas uma linguagem que pudesse combinar texto, imagens e botões em um navegador para a Internet, sem se preocupar com o intercâmbio de informações pela Web. O advento desta linguagem foi a grande responsável pela popularização da Internet, pois permitiu aos usuários de computadores de todo o mundo o acesso a um sem número de documentos e informações que podiam ser lidas em seus computadores, onde quer que estejam. Com o crescimento do uso da Web, começou-se a constatar as limitações da HTML. A sua falta de extensibilidade era um grande motivo de frustração para os desenvolvedores. De modo a tentar dar mais fôlego a ela o W3C (*World Wide Web Consortiun*) criou uma tecnologia de folhas de estilo para a HTML, as Cascading Style Sheets (CSS). No entanto, as medidas tomadas tiveram apenas o intuito paliativo, pois a necessidade de uma nova linguagem, padronizada, plenamente extensível e estruturalmente escrita era real (DEITEL *et al*, 2003).

HTML é uma linguagem de publicação de hipertexto na Web. É um formato não-proprietário baseado na SGML (*Standard Generalized Markup Language* – linguagem de marcação generalizada padrão), e pode ser criado e processado por uma grande quantidade de ferramentas, dos editores de texto que não permitem nenhum tipo de formatação até editores de texto sofisticados com muitos recursos de formatação (como cor de fonte e tamanho de caracteres). Atualmente todos os navegadores para Web suportam a exibição de documentos HTML (W3C).

No escopo da transferência eletrônica de dados a HTML não se mostra adequada para auxiliar em tal tarefa. A HTML veio inicialmente como uma rápida solução para a exibição de documentos em variáveis plataformas distintas do que para o acesso a uma diversidade de arquivos, formatos e dados. Os desenvolvedores perceberam que com esta linguagem não poderiam estruturar e descrever tipos diferentes de dados. Em contrapartida, a linguagem XML se mostra bastante eficaz no que se refere ao intercâmbio de dados (PITTS-MOULTIS e KIRK, 2000).

Segundo o W3C (2004), XML é um formato de texto muito simples e flexível derivado da SGML. Originalmente projetado para encontrar chaves em documentos eletrônicos em larga escala, XML também está se tornando um importante padrão para a troca de uma vasta variedade de dados na web.

Para Tidel (2003), HTML é uma linguagem de marcação para apresentação, legível e exibida por praticamente qualquer navegador Web moderno, enquanto que XML é uma linguagem de marcação de conteúdo, sem elementos de apresentação inerente ou embutido: apenas elementos de definição de conteúdo (TIDEL, 2003).

Linguagens como HTML e XML são ditas de “marcação” por utilizarem símbolos para descreverem seu próprio conteúdo. Essas linguagens usam marcas (*tags*) no início e no fim de cada item do documento. Tanto em HTML quanto em XML as *tags* são envolvidas por `< >` e normalmente são relativamente auto-explicativas. Em HTML, por exemplo, para especificar que um texto será exibido em negrito basta colocar no início do texto a *tag* `` e no final a *tag* ``. Neste caso é possível constatar que a *tag* de fechamento é igual a *tag* de abertura com a inclusão de uma barra após o símbolo `<` (PITTS-MOULTIS e KIRK, 2000).

Na seqüência temos um código escrito em HTML para apresentar o menu de um restaurante:

```
<H2>Restaurante Bom Paladar</H2>
<TABLE WIDTH="480" CELSPACING="1">
<TR><TD><H3>Aperitivos</H3></TD></TR>
<TR>
<TD>Calabresa acebolada</TD>
<TD>R$ 5.50</TD>
<TR>
<TD>Provolone à milanesa</TD>
<TD>R$ 10,00</TD>
<TR>
<TR><TD><H3>Especialidades da casa</H3></TD></TR>
<TR>
<TD>Prato feito</TD>
<TD>R$ 4,50</TD>
<TR>
<TD>Filé ao molho madeira</TD>
<TD>R$ 11,50</TD>
</TR>
```

A Figura 3 mostra como o código anterior será visualizado em um navegador.

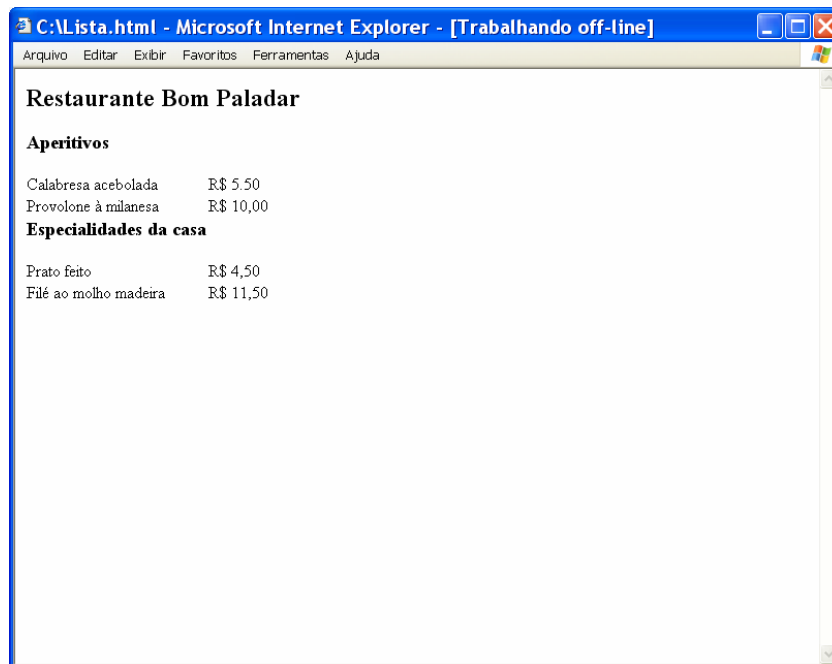


Figura 3: Código HTML exibido em um navegador.

Ao contrário da HTML, que possui um número limitado de *tags* com funções predefinidas, XML permite que sejam criadas *tags* próprias do usuário para fins específicos. Com esta propriedade da XML é possível criar *tags* que sejam facilmente entendidas pelas pessoas (DEITEL *et al*, 2003).

Para exemplificar tal afirmação o código em HTML anterior poderia ser escrito também em XML da seguinte forma:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Menu>
  <Estabelecimento>Restaurante Bom Paladas</Estabelecimento>
  <Grupos>
    <Grupo>
      <Titulo>Aperitivos</Titulo>
      <Itens>
        <Item>
          <Descricao>Calabresa acebolada</Descricao>
          <Preco>R$ 5,50</Preco>
        </Item>
        <Item>
          <Descricao>Provolone à milanesa</Descricao>
          <Preco>R$ 10,00</Preco>
        </Item>
      </Itens>
    </Grupo>
    <Grupo>
      <Titulo>Especialidades da casa</Titulo>
      <Itens>
        <Item>
          <Descricao>Prato feito</Descricao>
          <Preco>R$ 4,50</Preco>
        </Item>
      </Itens>
    </Grupo>
  </Grupos>
</Menu>
```

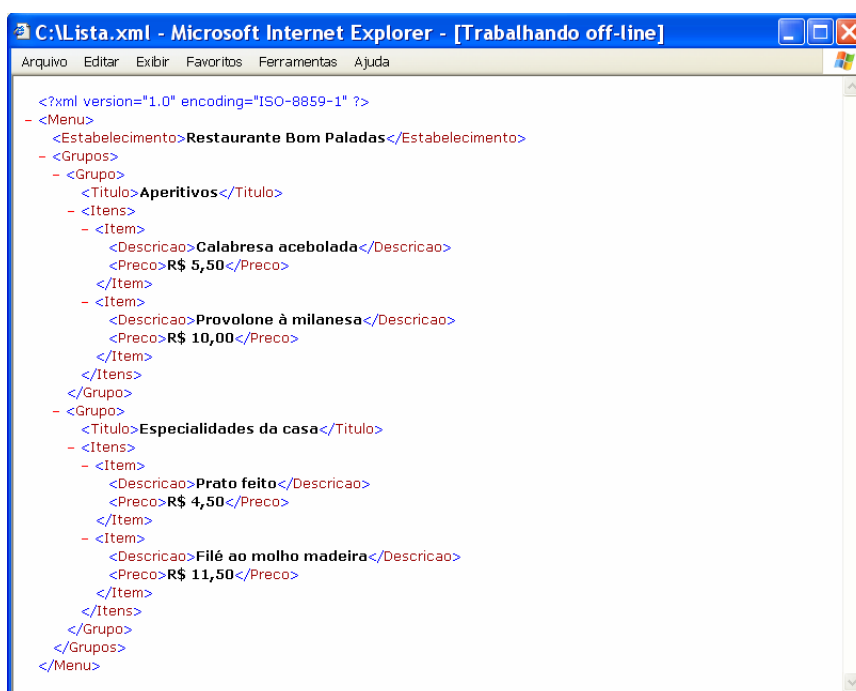
```

</Item>
<Item>
<Descricao>Filé ao molho madeira</Descricao>
<Preco>R$ 11,50</Preco>
</Item>
</Itens>
</Grupo>
</Grupos>
</Menu>

```

A declaração *encoding*, que pode ser vista no documento anterior estabelece o conjunto de caracteres utilizado no documento. Para documentos em português, que aceita os caracteres da língua bem como a acentuação, pode ser utilizado ISO-8859-1 (DÉCIO, 2000).

O código XML apresentado acima será exibido no navegador como uma estrutura de nós como pode ser constatado na Figura 4.



The screenshot shows a web browser window titled "C:\Lista.xml - Microsoft Internet Explorer - [Trabalhando off-line]". The address bar shows "Arquivo Editar Exibir Favoritos Ferramentas Ajuda". The main content area displays the following XML code:

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
<Menu>
<Estabelecimento>Restaurante Bom Paladas</Estabelecimento>
<Grupos>
<Grupo>
<Titulo>Aperitivos</Titulo>
<Itens>
<Item>
<Descricao>Calabresa acebolada</Descricao>
<Preco>R$ 5,50</Preco>
</Item>
<Item>
<Descricao>Provolone à milanese</Descricao>
<Preco>R$ 10,00</Preco>
</Item>
</Itens>
</Grupo>
<Grupo>
<Titulo>Especialidades da casa</Titulo>
<Itens>
<Item>
<Descricao>Prato feito</Descricao>
<Preco>R$ 4,50</Preco>
</Item>
<Item>
<Descricao>Filé ao molho madeira</Descricao>
<Preco>R$ 11,50</Preco>
</Item>
</Itens>
</Grupo>
</Grupos>
</Menu>

```

Figura 4: Código XML exibido em um navegador.

As marcações do exemplo anterior, como <Estabelecimento> e <Item>, são totalmente auto-explicativas não necessitando de qualquer informação adicional para que uma pessoa possa entender seu significado, mesmo que esta não tenha conhecimento da linguagem ou até mesmo de informática.

A linguagem XML, assim como outras linguagens de programação existentes (C++, Pascal, Java, por exemplo), possui termos reservados da própria linguagem. Esses termos podem ser utilizados apenas para o fim que é imposto pela linguagem. No caso de XML existem caracteres reservados que devem ter sua função respeitada na elaboração de um documento XML, mas caso seja necessário incluir estes caracteres no texto sem que estes sejam tomados como caracteres reservados, é possível substituí-los por uma expressão como pode ser visto na Tabela 3 (CANTÚ, 2003).

Caracter reservado	Expressão
<	<
>	>
&	&
'(apóstrofo)	&após;
“(aspas)	"

Tabela 3: Tabela de conversão de caracteres reservados.

Incluir comentários dentro de um documento XML é possível desde que sejam escritos entre as *tags* `<!--` e `-->`. O texto escrito entre elas é ignorado por qualquer processador de XML. No caso de ser necessário incluir em um documento XML algum conteúdo não-XML (por exemplo, informações binárias ou um *script*) que potencialmente podem conter caracteres reservados da linguagem, pode se fazer uso da seção CDATA, onde o texto deverá ser incluído entre as *tags* `<![CDATA[` e `]]>` (CANTÚ, 2003).

Cantù (2003) destaca ainda outro ponto importante sobre a estrutura da linguagem XML. O autor aponta que XML faz diferenciação entre letras maiúsculas e minúsculas (ao contrário da HTML), isto significa que a *tag* `<Estabelecimento>` é totalmente diferente de `<estabelecimento>` ou de `<ESTABELECIMENTO>`. Esta propriedade da linguagem implica que é preciso muita atenção nas definições de *tags* em um documento XML, pois uma *tag* utilizada na abertura de um item do documento deverá ser escrita exatamente igual quando o item for fechado.

Os usos iniciais de XML incluíam documentos para navegadores Web tradicionais e troca de documentos específicos. Setores, tais como químico,

matemática, imobiliário, observação meteorológica, bancário, intercâmbio eletrônico de dados (EDI) e muitos outros oferecem um vasto potencial de utilização de XML. Uma vez que uma DTD define o documento XML de modo que ele seja específico para as necessidades dos usuários, ela pode ser prontamente adaptada para qualquer tipo de aplicação (TIDEL, 2003).

Atualmente têm surgido diversas aplicações que fazem uso da linguagem XML, não só nas aplicações voltadas à Web, mas também em aplicações internas de uma empresa. Segundo Pitts-Moultis e Kirk (2000), “os aplicativos XML orientados a finanças e comércio abrem um novo mundo para consumidores, instituições financeiras e negócios”. Ainda para estes autores, estes aplicativos permitem que os consumidores e empresas possam utilizar uma linguagem padronizada para se comunicarem.

Pitts-Moultis e Kirk (2000) descrevem alguns aplicativos baseados em XML, todavia, a implementação do XML/EDI se enquadra perfeitamente no que se propõe neste projeto referente à exportação e importação de dados. O XML/EDI oferece aos fornecedores uma estrutura e formato padronizados para descrever diferentes tipos de dados usados no processamento de diversos tipos de transações, tais como, pedidos, pagamentos e cotações de preço.

Para MacGrath (1999), a iniciativa XML/EDI tem como objetivo unir a XML e o EDI enquanto retém as melhores características de cada um na perspectiva de comércio eletrônico. A XML tem muitas vantagens como formato de dados EDI, sendo que muito tempo e dinheiro vêm sendo gastos com a XML e seus especialistas. O principal objetivo do EDI – reduzir o custo do comércio – é beneficiado pelo formato aberto da XML, sua legibilidade, e o conceito de DTD (*Document Type Definition* – definição de tipo de documento) que será visto adiante.

Não se poderia falar da XML sem dar uma atenção especial à DTD. As DTD's são responsáveis por definir a estrutura de um documento. O documento XML não necessita ter uma DTD correspondente. Entretanto, o uso de DTD's se faz extremamente interessante para garantir a conformidade do documento,

especialmente em *transaction business-to-business (B2B)*, nas quais dois documentos XML são intercambiados (DEITEL *et al*, 2003).

Complementando a definição anterior, Pitts-Moultis e Kirk (2000) explicam que “uma DTD define as partes de um documento e descreve como eles podem ou não serem usados, o que pode ser colocado em seus interiores e se são ou não elementos obrigatórios do documento”. Ainda para os autores uma DTD pode incluir declarações de elementos e atributos, as entidades, notações e comentários que serão permitidos.

Existem três formas de classificar um documento XML: inválido, válido e bem-formado. Um documento XML será considerado inválido caso algum analisador sintático acuse alguma não conformidade com as definições da linguagem (como por exemplo, uma *tag* não fechada); um documento será classificado como válido no caso estar em coerente com a DTD especificada no cabeçalho do arquivo XML; por fim, um documento será considerado bem-formado quando estiver sintaticamente correto, mas não estiver de acordo com as regras da DTD (DEITEL *et al*, 2003).

Tomando como base o código XML representado na Figura 4, pode-se elaborar uma DTD para garantir que o documento XML seja um documento válido. O código a seguir representa a DTD do referido exemplo:

```
<!ELEMENT Menu (Estabelecimento ,Grupos ) >  
<!ELEMENT Estabelecimento (#PCDATA) >  
<!ELEMENT Grupos (Grupo*) >  
<!ELEMENT Grupo (Titulo ,Itens ) >  
<!ELEMENT Titulo (#PCDATA) >  
<!ELEMENT Itens (Item*) >  
<!ELEMENT Item (Descricao ,Preco ) >  
<!ELEMENT Descricao (#PCDATA) >  
<!ELEMENT Preco (#PCDATA) >
```

Uma DTD pode ser armazenada tanto interna quanto externamente a um documento XML. Elaborar um documento XML contendo uma DTD interna indica que o documento será “auto-suficiente”, ou seja, quando o documento XML for processado não será necessário acessar outros arquivos para validar a sua estrutura. Já um documento XML com uma declaração de DTD externa implicará na leitura de outro arquivo para que o documento seja validado. No escopo deste

projeto será dada atenção a DTD's definidas externamente. Ainda podem ser utilizadas DTD's disponíveis publicamente que já tenham sido definidas para uma determinada necessidade ou utilizar DTD's elaboradas localmente. As DTD's definidas publicamente não farão parte do contexto do projeto em questão (PITTS-MOULTIS e KIRK, 2000).

Para que um documento XML seja validado por uma DTD externa é necessário incluir em seu cabeçalho a referência para a sua DTD. Na seqüência poderá ser verificada a alteração no documento XML para considerar a DTD elaborada anteriormente:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE Estabelecimento SYSTEM "Lista.dtd">
<Menu>
  <Estabelecimento>Restaurante Bom Paladas</Estabelecimento>
  <Grupos>
    <Grupo>
      <Titulo>Aperitivos</Titulo>
      <Itens>
        <Item>
          <Descricao>Calabresa acebolada</Descricao>
          <Preco>R$ 5,50</Preco>
        </Item>
        <Item>
          <Descricao>Provolone à milanesa</Descricao>
          <Preco>R$ 10,00</Preco>
        </Item>
      </Itens>
    </Grupo>
    <Grupo>
      <Titulo>Especialidades da casa</Titulo>
      <Itens>
        <Item>
          <Descricao>Prato feito</Descricao>
          <Preco>R$ 4,50</Preco>
        </Item>
        <Item>
          <Descricao>Filé ao molho madeira</Descricao>
          <Preco>R$ 11,50</Preco>
        </Item>
      </Itens>
    </Grupo>
  </Grupos>
</Menu>
```

Pode-se perceber duas declarações diferentes dos documentos XML apresentados anteriormente: *standalone* e *DOCTYPE*. A opção *standalone* no cabeçalho do documento XML indica se o documento depende de uma DTD externa ou não (quando depender de um DTD externo a opção deverá ser igual a "no" e

quanto não depender deverá ser igual a “yes”). A declaração *DOCTYPE* especifica a DTD responsável pela validação do documento, sendo que se a DTD for interna no documento XML a estrutura da DTD deverá vir após esta declaração e, se for uma DTD externa virá após esta declaração o nome do arquivo com extensão “.dtd”, conforme apresentado no exemplo anterior.

Cantù (2003), no que se refere à “manipulação” dos documentos XML, destaca duas técnicas normais para a manipulação de documentos XML. A primeira seria usando uma *interface* DOM (*Document Object Model* – modelo de objetos documento), e a segunda usando SAX (*Simple API for XML* – API simples para XML). As duas estratégias são bastante distintas:

- O DOM carrega o documento inteiro em uma árvore hierárquica de nós na memória, permitindo que você os leia e manipule-os para alterar o documento. Por isso, o DOM é conveniente quando você deseja navegar na estrutura XML na memória e editá-la, ou para a criação de documentos desde o início.
- O SAX analisa o documento, chamando um evento para cada elemento do documento, sem fazer uso da memória para construção de qualquer tipo de estrutura. Uma vez analisado pelo SAX, o documento é perdido, mas essa operação geralmente é muito mais rápida do que a construção de árvores do DOM. Usar o SAX é bom para ler um documento – por exemplo, quando se está à procura de parte de seus dados.

DOM e SAX são conhecidos como analisadores sintáticos de XML (*parsers*), que têm a capacidade ler o documento XML, verificar sua sintaxe (levando em consideração as regras definidas pela linguagem), relatar quaisquer erros e permitir, via programas, o acesso ao conteúdo do documento. Todavia, a verificação da sintaxe de um documento XML pode ser feita pelo *parser* sem que seja necessário processar documento.

Cantù (2003) apresenta ainda uma terceira maneira clássica de ler e criar documentos XML, esta técnica seria por meio de gerenciamento de *strings*. A

operação mais rápida é criar um documento pela adição de *strings*, particularmente se for possível realizar uma única passagem (e não precisar modificar nós já gerados). Mesmo a leitura de documentos por meio de funções de *string* é muito rápida, mas esse processo pode se tornar muito complicado em estruturas complexas, pois será necessário que o programa que está realizando a leitura faça a validação da estrutura do documento.

Quando for necessária a leitura de documentos XML muito grandes um *parser* do tipo SAX se mostra mais eficiente, pois, ao contrário de um *parser* DOM, este não cria nenhuma estrutura na memória, não necessitando assim de uma grande quantidade deste tipo de recurso. Já a utilização do DOM se torna mais interessante quando se deseja criar um documento XML ou fazer alterações em um arquivo já existente, pois o DOM fará as validações pertinentes a cada alteração na estrutura. Com isto é possível constatar que a escolha da técnica de manipulação de um documento XML, visando um melhor desempenho, deverá levar em consideração os tamanhos estimados dos documentos e o tipo de operação que será realizada sobre ele. Isto quer dizer que um mesmo programa poderá ter manipulações de documentos XML com uma, duas e até as três técnicas referidas anteriormente (CANTÙ, 2003).

2.4 Dll (Dynamic Link Library) ActiveX

Segundo a Microsoft (1997), “um “componente” *ActiveX* é uma unidade de código executável como, por exemplo, um arquivo .exe, .dll ou .ocx, que segue a especificação *ActiveX* para a criação de objetos. A tecnologia *ActiveX* permite ao programadores montar esses componentes de software reutilizáveis em aplicativos e serviços.

Já Brown (1999) descreve os componentes *ActiveX* como sendo parte do modelo de objeto componente (COM, *Component Object Model*) da Microsoft. Os componentes COM, conforme aponta o autor, podem ser acessados de uma única

forma, quer estejam sendo executados na mesma máquina, em máquinas separadas com sistemas operacionais diferentes ou em máquinas separadas com hardware e sistema operacional diferentes. O modelo de extensibilidade do *ActiveX* impressiona, pois permite desenvolver aplicativos que podem ser executados de forma independente, em rede ou via internet, como se estivessem em seu computador. O *ActiveX* faz o ajuste do COM de maneira ordenada em uma *interface* fácil de usar e igualmente consistente em sistemas similares e diferentes.

Um objeto COM pode conter mais de uma *interface* de objeto, onde cada uma tem um nome. A *interface* isola o objeto COM dos detalhes de implementação. O usuário da *interface* (seja um outro objeto COM ou um programa executável, por exemplo) não precisa conhecer os detalhes de implementação, o usuário deve se preocupar apenas em como a *interface* trabalha e não sobre os detalhes atrás da *interface*. Isolando a *interface* de sua implementação pode-se construir componentes que podem ser substituídos e reutilizados. Isto simplifica e pode multiplicar a utilidade do objeto COM. Uma *interface* por sua vez pode possuir várias funções e procedimentos onde serão implementadas as funcionalidades do objeto COM, essas funções e procedimentos também são conhecidos como métodos (Microsoft, 1997).

A Figura 5 a seguir mostra a estrutura de um objeto COM.

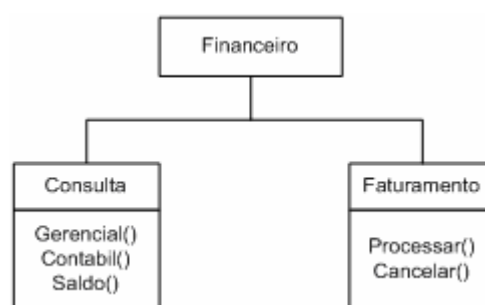


Figura 5: Exemplo de um objeto COM.

O objeto de nome “Financeiro” possui duas *interfaces* de objeto: Consulta e Faturamento. A *interface* “Consulta” com três métodos: Gerencial, Contábil e Saldo. A *interface* “Faturamento” possui dois métodos: Processar e Cancelar. A codificação dos processos é feito nesses métodos. Um método pode, opcionalmente, receber parâmetros e pode ou não retornar algum valor. (BROWN, 1999).

Tanto para um objeto COM quanto para um componente *ActiveX* não é possível referenciar mais de uma *interface* ao mesmo tempo, ou seja, se for necessário executar os processos das duas *interfaces* representadas na Figura 5 serão necessárias duas referências para o objeto Financeiro (MICROSOFT, 1997).

```
Procedure Teste;  
Var  
  FinanceiroConsulta, FinanceiroFaturamento: Variant;  
Begin  
  FinanceiroConsulta := CreateOleObject('Financeiro.Consulta');  
  FinanceiroConsulta.Gerencial;  
  
  FinanceiroFaturamento := CreateOleObject('Financeiro.Faturamento');  
  FinanceiroFaturamento.Processar;  
End;
```

O código acima, escrito em *Object Pascal*, mostra um exemplo de como executar os métodos do objeto “Financeiro”. A função *CreateOleObject* cria o objeto e passa para uma variável a sua referência de memória para que os seus métodos possam ser executados posteriormente, como exemplificado pelas variáveis “FinanceiroConsulta” e “FinanceiroFaturamento”. A referida função possui apenas um parâmetro (obrigatório) onde deve ser passado o nome do objeto e o nome da *interface* separado com um ponto final (CANTÙ, 2003).

O desenvolvimento de componentes utilizando a tecnologia *ActiveX* não deve ser confundido com programação orientada a objetos (POO). A POO é uma forma de criar componentes baseados em objetos, enquanto a *ActiveX* é uma tecnologia que permite a combinação de componentes baseados em objeto, criados em muitas ferramentas diferentes. Resumindo, *ActiveX* consiste em fazer os objetos colaborarem entre si (MICROSOFT, 1997).

Sendo assim é possível construir uma gama de objetos em uma linguagem orientada a objetos (como o *Object Pascal* utilizado pelo Delphi da Borland) e agrupá-los dentro de componentes *ActiveX*. Tendo feito esta junção pode-se utilizá-los em qualquer ferramenta de programação que suporte a tecnologia *ActiveX*.

2.5 Bancos de Dados e Acesso

Levando-se em consideração que o objetivo do projeto é a transferência eletrônica de dados, é mais do que lógico pensar na origem e no destino destes dados. O principal objetivo neste aspecto é a transferência entre bancos de dados distintos. Existem vários tipos de modelos de banco de dados: hierárquico, de rede, relacional e orientado a objetos. No escopo deste trabalho foi dado enfoque ao modelo relacional, devido ser este o modelo mais utilizado atualmente.

Os bancos de dados atuais são gerenciados pelo Sistema Gerenciador de Banco de Dados (SGBD), também conhecidos como *Data Base Manager System* (DBMS). Um SGBD consiste em uma coleção de dados inter-relacionados e em um conjunto de programas para acessá-los. A finalidade principal de um SGBD é o de proporcionar um ambiente onde os dados possam ser armazenados e recuperados. O trabalho de um SGBD consiste em propiciar, de forma ordenada, o acesso aos dados por parte das aplicações que necessitam destes dados. Outras funções também são delegadas aos gerenciadores como: controle de segurança de acesso, integridade dos dados, controle de concorrência, execução e recuperação de cópias de segurança (KORTH e SILBERSCHATZ, 1993).

O ODBC (*Open Database Connectivity*) é uma *interface-conector* que foi criada para se obter uma padronização de conexões aos diferentes gerenciadores de bancos de dados. O *driver* ODBC fará a tradução dos comandos padronizados da *interface* ODBC para os do gerenciador de banco de dados. A ponte entre um banco de dados, ou seu gerenciador, e a *interface* de programação é feita através do *Data Source Name* (DSN), que será o identificador desse banco dentro da linguagem de programação.

Para criar um DSN é necessário acessar o aplicativo do Windows “Fonte de Dados (ODBC)” e solicitar a criação um uma nova fonte de dados. Será preciso escolher o *driver* de acesso correspondente ao SGBD a ser acessado (como SQL Server, Oracle ou DB2) e informar os dados de conexão particulares de cada SGBD.

A Figura 6 apresenta a tela de configuração de um DSN para Oracle.

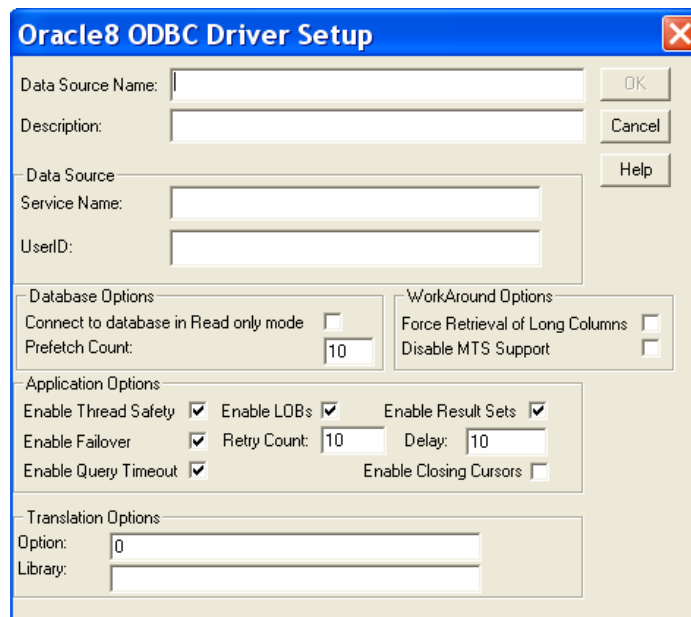


Figura 6: Tela de configuração de uma conexão ODBC para Oracle.

As informações requeridas para a configuração de um DSN irão variar conforme o gerenciador de banco de dados. A Figura 7 ilustra essa variação exibindo as informações necessárias para a elaboração de um DSN para MS-Access.

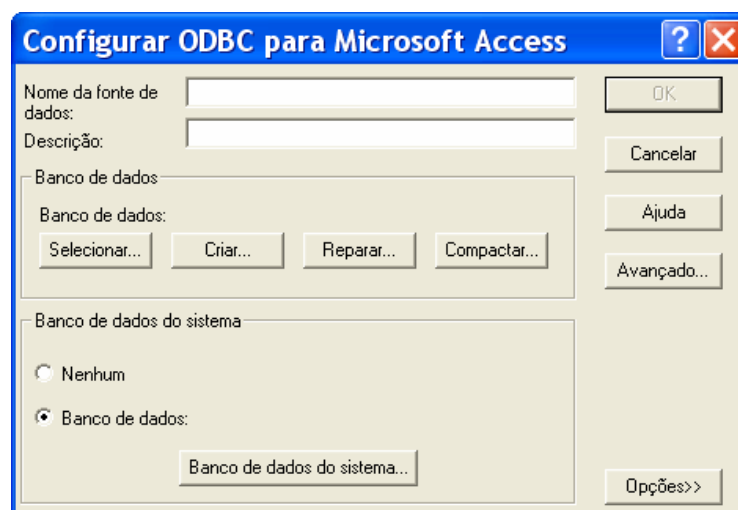


Figura 7: Tela de configuração de uma conexão ODBC para MS-Access.

O ADO (*Activex Data Objects*) é a interface de alto nível da Microsoft para acesso a dados. Sua implementação se baseia na tecnologia de acesso a dados

OLE DB da Microsoft, que fornece acesso a bancos de dados relacionais e não relacionais, e a outros objetos e sistemas de arquivo que não interessam ao foco do projeto. A configuração do acesso utilizando ADO se faz criando uma *string* de conexão contendo os dados de conexão ao SGBD que se deseja acessar. Esta *string* de conexão deverá ser informada quando o *driver* ADO for acionado (CANTÙ, 2003).

Como exemplo de uma *string* de conexão para ADO utilizaremos o SGBD MS SQL Server, um servidor denominado SERVER, um banco de dados nomeado de DBPROJETO e usuário denominado USUARIO. Como resultado se terá a seguinte *string* de conexão:

```
Provider=SQLOLEDB.1;User ID=USUARIO;Initial Catalog=DBPROJETO;Data Source=SERVER
```

Para uma conexão com um SGBD Oracle teríamos uma *string* de conexão como a que será apresentada a seguir. A conexão se baseará em um servidor de nome SERVER e um usuário denominado USUARIO:

```
Provider=MSDAORA.1;User ID=USUARIO;Data Source=SERVER
```

Assim como pôde ser constatado com a configuração de um DSN, uma conexão ADO também terá parâmetros diferenciados conforme o SGBD com o qual se deseja estabelecer uma conexão.

2.6 Ferramenta de Programação

O Delphi é uma ferramenta de programação da empresa Borland. Suas raízes estão na linguagem Pascal, difundida no meio acadêmico para iniciar os estudantes na programação estruturada. O Turbo Pascal surgiu como uma evolução, não tanto da linguagem, mas no que se refere ao ambiente de desenvolvimento. Criou-se então um ambiente onde, numa mesma *interface* visual, os programas pudessem ser codificados, compilados e depurados com facilidade. A Borland procurou

reaproveitar o conceito de IDE (*Integrated Development Environment*) do antigo Turbo Pascal, adicionado a isto a evolução do Pascal para o *Object Pascal*, já no conceito de programação orientada a objetos (POO) (CANTÙ, 2003).

Conforme descreve Cantù (2003), o Delphi possui um componente que implementa as funcionalidades de um *parser* baseado em DOM (componente TXMLDocument) e, embora não inclua um suporte específico à *interface* SAX, pode-se facilmente incorporar o suporte XML da Microsoft, a biblioteca MSXML. A biblioteca MSXML é instalada nas versões mais recentes do Sistema Operacional Windows.

A Figura 8 mostra o componente TXMLDocument na barra de ferramentas do Delphi.



Figura 8: Componente TXMLDocument.

O componente TXMLDocument pode funcionar com algumas implementações de DOM: o MSXML da Microsoft, o Xerces da Apache Foundation e o OpenXML de código-fonte aberto. O OpenXML é escrito em *Object Pascal* e portanto não necessita de uma biblioteca externa para o programa ser executado (CANTÙ, 2003).

Tomando como base o documento XML a seguir, será apresentado um trecho de código-fonte *Object Pascal* para ler o documento.

```
<?xml version="1.0"?>
<Bancos>
  <Banco Identificador="BANCOSISTEMA">
    <Tipo>MSSQLServer</Tipo>
    <Servidor>SERVERTESTE</Servidor>
    <User>USUARIO</User>
  </Banco>

  <Banco Identificador="BANCOSISTEMA">
    <Tipo>MSSQLServer</Tipo>
    <Servidor>SERVERTESTE2</Servidor>
    <User>USUARIO</User>
  </Banco>
</Bancos>
```

O código fonte apresentado na seqüência utiliza o componente `TXMLDocument` para processar o documento XML referido anteriormente.

```

Procedure TesteXMLDOM;
Var
  i, z: Integer;
  vIdentificador,
  vTipo,
  vServidor,
  vUser: variant;
  vNode1, vNode2: IXMLNode;
  XMLDocumento: TXMLDocument;
Begin
  XMLDocumento:= TXMLDocument.Create(nil);
  XMLDocumento.Active:= True;
  For i:= 1 to XMLDocumento.ChildNodes.Count do
  Begin
    vNode1:= XMLDocumento.DocumentElement.ChildNodes[i];
    vIdentificador:= vNode1.Attributes['Identificador'];
    For z:= 0 to vNode1.ChildNodes.Count -1 do
    Begin
      vNode2:= vNode1.ChildNodes[z];

      if vNode2.NodeName = 'Tipo' then
        vTipo:= vNode2.NodeValue
      else
        if vNode2.NodeName = 'Servidor' then
          vServidor:= vNode2.NodeValue
        else
          if vNode2.NodeName = 'User' then
            vUser:= vNode2.NodeValue;
          End;
        End;
      End;
    End;
  End;
End;

```

No exemplo acima, a variável `XMLDocumento` foi definida como sendo do tipo `TXMLDocument` para executar as funcionalidades do *parser* DOM. Devido ao *parser* do tipo DOM trabalhar com um documento XML montando uma estrutura de árvores em memória, foi necessário utilizar variáveis para armazenar os nós gerados pelo componente. As variáveis `vNode1` e `vNode2` são as responsáveis por armazenar a estrutura de um nó. O comando *For* é utilizado para percorrer a estrutura do documento nó por nó, e a propriedade *Attributes* permite que sejam acessados os atributos de um nó, sendo necessário informar o nome do atributo. Para acessar os elementos de um nó deve-se utilizar os atributos `NodeName` e `NodeValue` que, respectivamente, representam o nome do elemento e o valor do elemento.

Em relação aos componentes *ActiveX*, o Delphi possui suporte para se trabalhar com esta tecnologia, tanto pelo lado da criação de um novo componente quanto na utilização deste componente em programas escritos em *Object Pascal*. A

criação de componentes pode ser feita utilizando ferramentas de apoio à criação (denominados *wizards*). A Figura 9 mostra as opções do Delphi para criação de componentes COM.

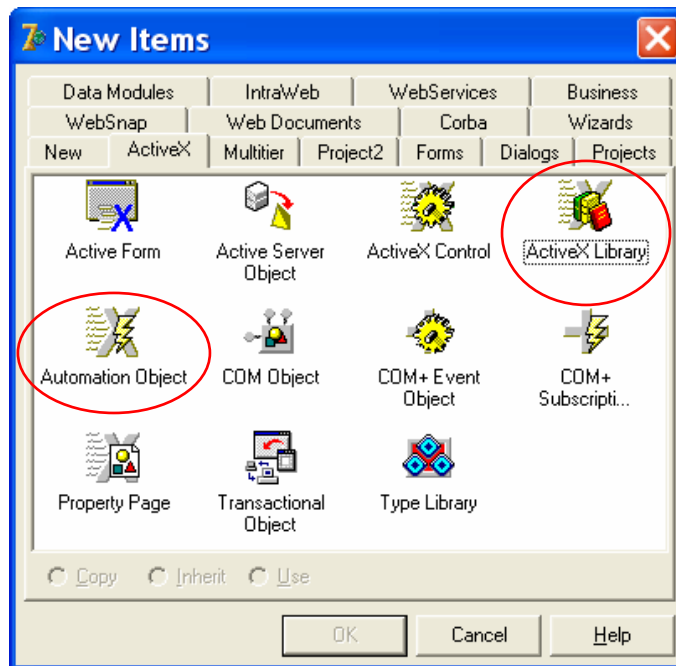


Figura 9: *Wizards* para criação de componentes *ActiveX*.

Para a criação de uma DLL *ActiveX* no Delphi deverá ser selecionada a opção *ActiveX Library* como pode ser visto na Figura 9. Para a criação de uma *interface* de objeto para a DLL será necessário selecionar o item *Automation Object*. Após selecionar o item *Automation Object* será exibida uma tela de *wizard* para auxiliar na sua criação. A Figura 10 mostra o *wizard* de criação da *interface* “Consulta”.

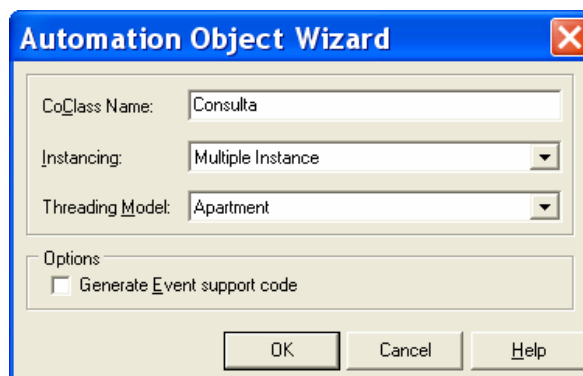


Figura 10: *Wizard* para a criação de interfaces de um componente *ActiveX*.

A estrutura de uma DLL *ActiveX* pode ser verificada na Figura 11. A figura

exibe um DLL de nome “Financeiro” com as *interfaces* “Consulta” (que possui os métodos Gerencial, Contábil e Saldo) e “Faturamento” (que contém os métodos Processar e Cancelar).

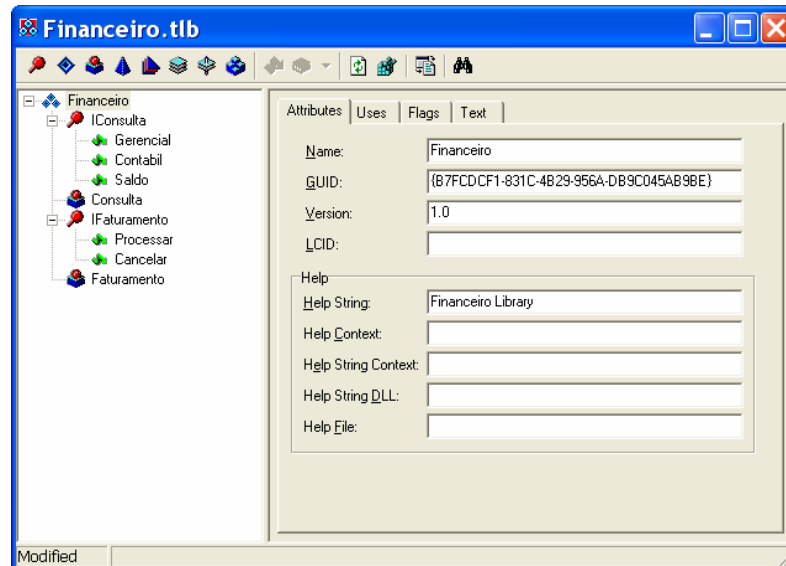


Figura 11: Exibição da estrutura de um componente *ActiveX* no Delphi.

Em relação ao acesso a dados, o Delphi possui suporte a algumas tecnologias de acesso, como ODBC, dbExpress (nova tecnologia implementada pela Borland) e tecnologias para acesso a dados remotos. Todavia, possui também um bom suporte ao ADO, com vários componentes que facilitam a conexão com os bancos e a manipulação dos dados (CANTÙ, 2003).

Na Figura 12 a seguir pode-se visualizar o componente TADOConnection utilizado para acessar SGBD's através do ADO.

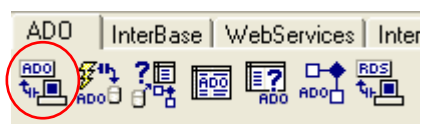


Figura 12: Componente TADOConnection.

O componente TADOConnection pode ser configurado para conectar SGBD's utilizando *drivers* ADO ou utilizando *drivers* ODBC. A configuração do componente pode ser feita visualmente através de uma tela exibida na Figura 13.

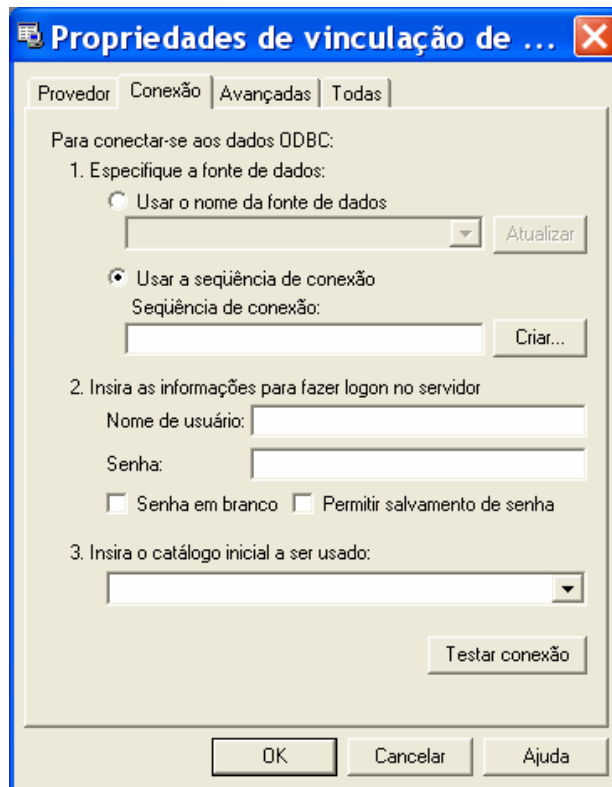


Figura 13: Tela de configuração do componente TADOConnection.

Para configurar o componente para acesso via ODBC deverá ser selecionada a opção “Usar o nome da fonte de dados” e escolher uma fonte de dados ODBC previamente configurada. Escolhendo a opção “Usar a seqüência de conexão” permitirá especificar uma *string* de conexão para acesso via ADO. Opcionalmente poderão ser configurados o nome de usuário e a senha (CANTÙ, 2003).

Outra forma de utilizar o componente TADOConnection é configurá-lo em tempo de execução, alimentando seus parâmetros para acesso. O código-fonte a seguir demonstra um exemplo de como o componente pode ser configurado durante a execução do programa.

```

Procedure TesteConexao;
Var
  ADOConnection: TADOConnection;
Begin
  ADOConnection := TADOConnection.Create(nil);
  ADOConnection.ConnectionString:= 'Provider=MSDAORA.1;User ID=USUARIO;Data' +
    'Source=SERVER'
  ADOConnection.Connected := True;
End;

```

O exemplo anterior demonstra a conexão via ADO a um SGBD Oracle. A propriedade "ConnectionString" é alimentada com a *string* de conexão. Para indicar que a conexão deve ser estabelecida deverá ser configurada a propriedade "Connected" com o valor "True", e para indicar que a conexão deve ser desfeita a mesma propriedade deve ser configurada com o valor "False".

CAPÍTULO III – FERRAMENTA PROPOSTA

3.1 Apresentação

Neste capítulo será apresentada a ferramenta Exchange Data XML (EDX). Inicialmente serão apresentadas as funcionalidades da ferramenta, descrevendo a maneira como a mesma deverá ser operada. Finalizando o capítulo serão apresentados os detalhes de implementação e a modelagem UML utilizada no desenvolvimento do projeto.

EDX é uma ferramenta de migração, exportação e importação de dados que visa facilitar a configuração e execução desses processos. A ferramenta deverá ser utilizada por empresas que necessitem fazer uma migração de sistemas ou necessitem trocar dados (como pedidos de compra, notas fiscais, acompanhamento de cargas) com outras empresas.

3.2 Funcionalidades

A ferramenta EDX possui dois grupos de funcionalidades. O primeiro grupo é responsável pela configuração (inclusão, alteração e exclusão) de todos os modelos utilizados pela ferramenta. O segundo grupo é responsável por executar os processos de transferência de dados suportados pela ferramenta.

Para que os processos de transferência possam ser executados é necessário alimentar alguns cadastros. Existem cinco cadastros que devem ser alimentados para a execução dos processos acima:

- Cadastro de Conexões: Este cadastro armazena modelos de parâmetros

para estabelecer conexões com bancos de dados. É possível incluir vários parâmetros de conexão que poderão ser utilizados nos modelos de transferência de dados (importação, exportação e migração). Este cadastro é imprescindível para o funcionamento da ferramenta, pois todas as conexões serão feitas com base nos modelos de parâmetros aqui cadastrados. A Figura 14 exhibe a tela de cadastramento dos modelos de conexão.

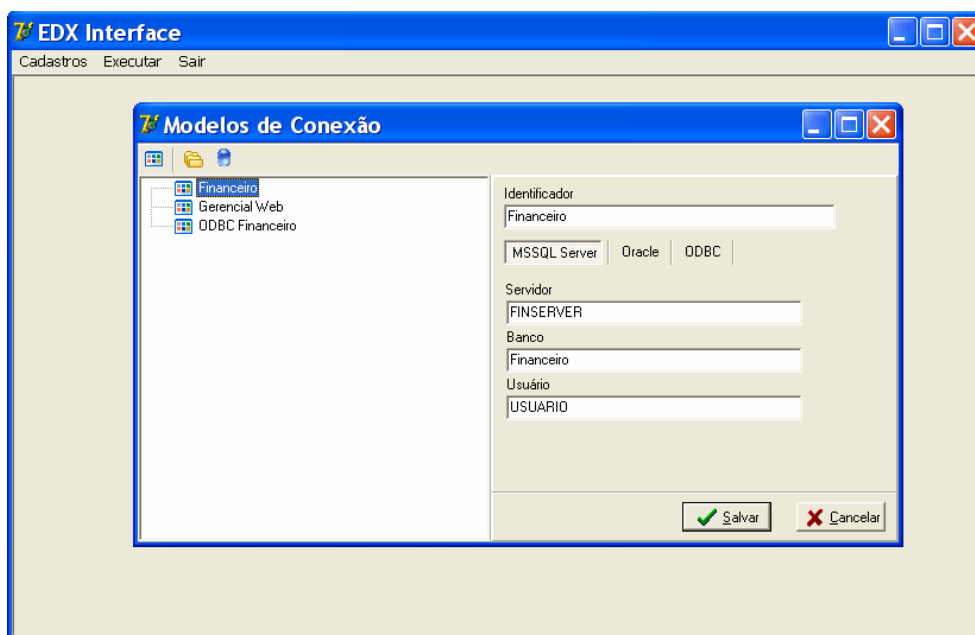



Figura 14: Tela de cadastro dos modelos de conexão.



Todas as telas de cadastro da ferramenta possuem a mesma *interface* visual padrão. No alto à esquerda encontra-se a barra de ferramentas, que possui ícones responsáveis pela inclusão de um novo modelo e de seus itens, ou ainda as funções de duplicação e exclusão do modelo. No canto esquerdo existe uma área que exhibe a estrutura dos modelos cadastrados de maneira semelhante a uma estrutura de árvore (denominada *tree-view*). Do lado direito da tela são solicitadas as informações necessárias para configurar cada item do modelo, onde também se encontram dois botões: Salvar (responsável por incluir ou atualizar o item no modelo) e Cancelar (responsável por cancelar a inclusão ou alteração de um item do modelo).

No cadastro do modelo de conexões o ícone  tem a função de inserir um novo modelo de parâmetro de conexão. Ao selecionar tal opção será necessário

informar os seguintes dados:

- Identificador: Nome do modelo de conexão. Neste modelo, e também em nenhum outro, será permitido cadastrar dois modelos com o mesmo identificador. Isto também é verdadeiro para itens dos modelos que possuam o item “Identificador”.

- Tipo da conexão: Será permitido escolher entre três tipos de conexão, cada uma com seus respectivos dados de conexão. Para conexão com bancos MSSQL Server será necessário informar o nome do servidor, o nome do banco a ser acessado e o usuário. Em conexões com Oracle deverão ser preenchidos o nome do servidor e o usuário. No caso da escolha ser uma conexão via ODBC será necessário informar o DSN correspondente à conexão desejada.

Os ícones  e  possuem, respectivamente, as funções de duplicar um modelo existente e excluir um modelo ou item de um modelo. Estes ícones são padrões e estão presentes em todas as telas de cadastramento da ferramenta.

- Cadastro de Tabelas de Conversão: As tabelas de conversão são utilizadas quando for necessário converter um determinado valor em outro. Todas as modalidades de transferência de dados podem fazer uso de uma tabela de conversão, desde que devidamente identificada no modelo correspondente.

Na Figura 15 é possível visualizar a tela de cadastro das tabelas de conversão.

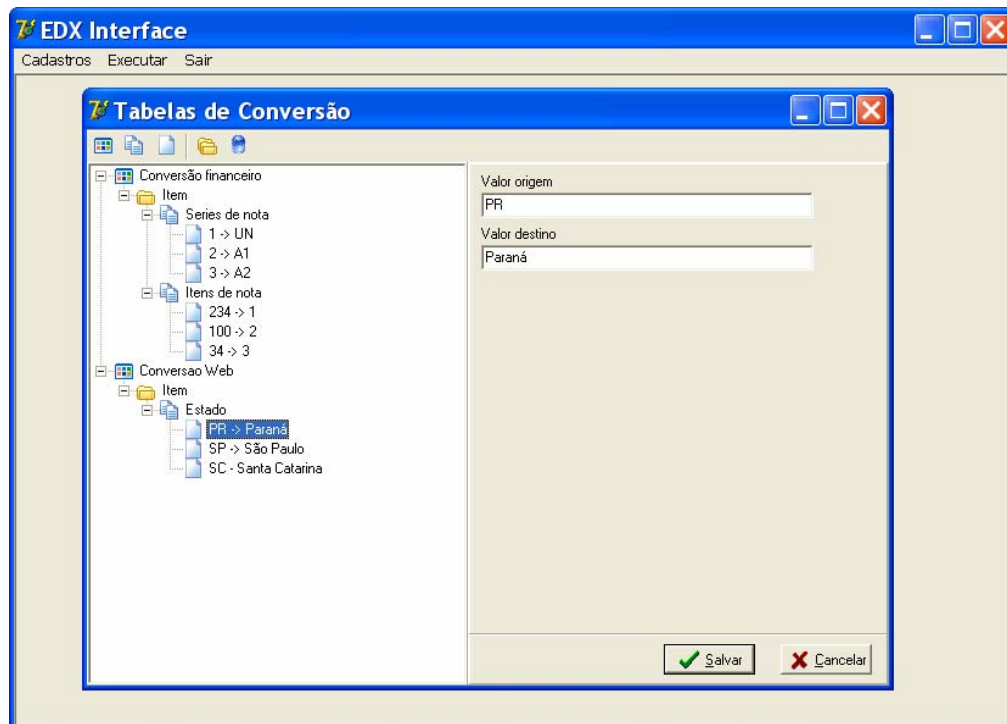





Figura 15: Tela de cadastro das tabelas de conversão.

Quando um novo modelo é inserido através do ícone  será exigido que seja informado o identificador do modelo. Não serão permitidas duas tabelas de conversão com o mesmo identificador.

O ícone  é responsável por solicitar a inclusão de um novo item de conversão onde será exigido o seu identificador, também único, contudo apenas dentro do modelo. Para incluir um registro de conversão deverá ser acionado o ícone  que solicitará que sejam informados os valores de origem e de destino, não sendo permitido dois valores de origem iguais dentro de um mesmo item de conversão.

- Cadastro de Modelos de Importação: Necessário para permitir a execução do processo de importação.
- Cadastro de Modelos de Exportação: Necessário para a execução do processo de exportação de dados.
- Cadastro de Modelos de Migração: O processo de migração de dados exige que seja cadastrado um modelo de migração.

As Figuras 16, 17 e 18 mostram, respectivamente, como os modelos de importação, exportação e migração são cadastrados.

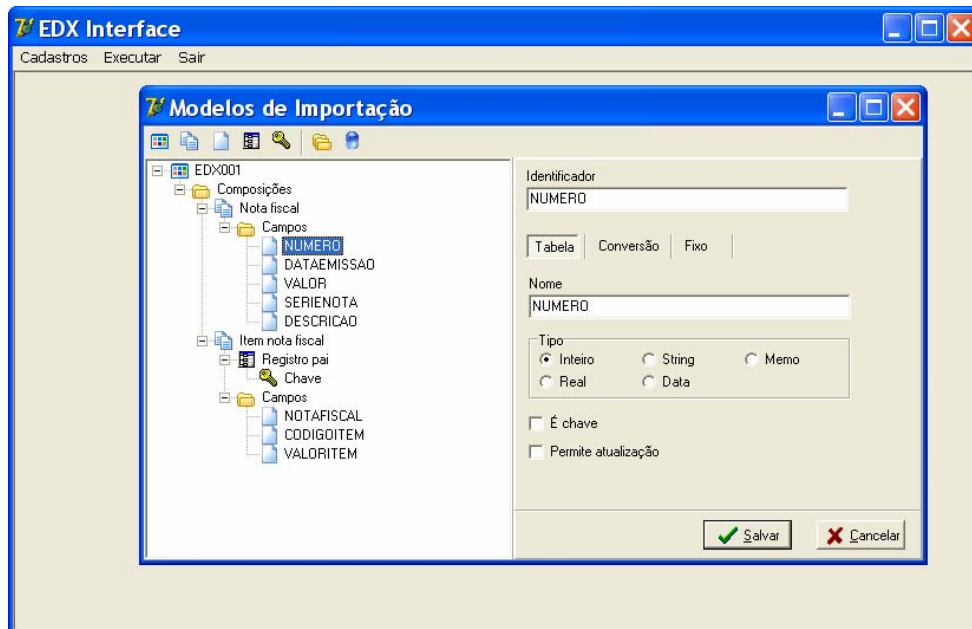


Figura 16: Tela de cadastro dos modelos de importação.

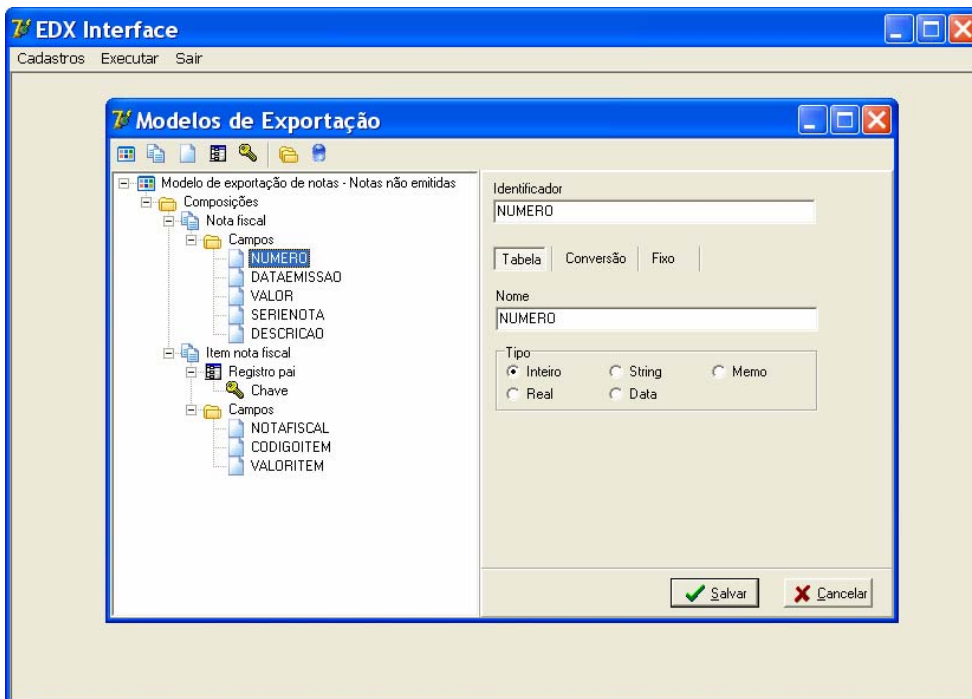


Figura 17: Tela de cadastro dos modelos de exportação.

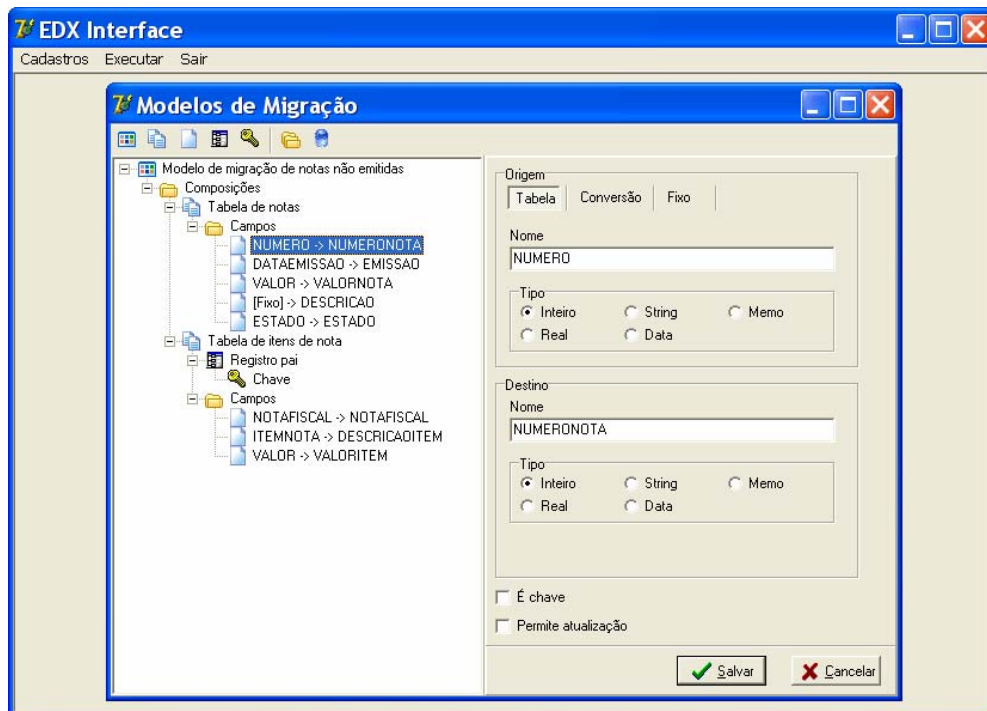




Figura 18: Tela de cadastro dos modelos de migração.

As telas de cadastrado dos modelos de importação (Figura 16), exportação (Figura 17) e migração (Figura 18) são semelhantes e possuem os mesmos ícones de funções. O ícone  aciona a inclusão de um novo modelo. Para tal função será exigido informar:


- Código de intercâmbio (apenas para importação e exportação): Não será permitido cadastrar dois modelos com o mesmo código de intercâmbio (apenas importação). No arquivo de importação existe um item para esta informação que será utilizada para identificar qual será o modelo de importação a ser utilizado no processo. No processo de exportação o arquivo XML a ser criado terá um item com tal informação.
- Identificador (apenas para exportação e migração): Nome do modelo.
- Conexão: Será permitido selecionar um dos modelos de conexão cadastrados para ser utilizado no processo de importação quando for necessário estabelecer a conexão com o banco de dados. Para

os modelos de migração será necessário selecionar os modelos de conexão para origem e destino.

- Tabela de conversão: Informar qual será a tabela de conversão a ser utilizada. Esta tabela deverá estar previamente cadastrada.
- Descrição (apenas importação): Esta informação é utilizada apenas para se poder caracterizar melhor o modelo, não tendo função nenhuma além de cadastral.



Quando o ícone  for acionado será incluída uma nova composição (conjunto de informações relativas a tabelas e campos) onde serão exigidos os seguintes dados:

- Identificador: Nome da composição. Este identificador deve ser único dentro do modelo.
- Tabela de origem (apenas exportação e migração): Nome da tabela de origem dentro do banco de dados de onde os dados serão extraídos.
- Filtro (apenas exportação e migração): Instrução SQL a ser utilizada para filtrar os registros a serem selecionados.
- Tabela de destino: Nome da tabela onde os dados serão atualizados.

Para incluir um campo dentro de uma composição será preciso acionar o ícone  onde serão exigidas as informações relativas aos campos das tabelas:

- Identificador (apenas importação e exportação): Identifica um item da composição relativo aos campos de uma tabela. Para a importação este item possui as informações de em qual a informação do arquivo será atualizada. Já para a exportação as informações são relativas ao campo de origem dos dados.

- Tipo: Em todos os modelos será necessário informar o tipo do dado (que pode assumir: Inteiro, Data, Real, String ou Memo) do campo, no entanto, no modelo de migração será necessário informar o tipo para a origem e para o destino. Um campo também poderá ter um valor fixo ou estar vinculado a uma tabela de conversão, onde o valor de origem será convertido antes de ser enviado para o destino.
- É chave: Indica se o campo faz parte do índice de busca para verificar se o registro já existe.
- Permite atualização: No caso do registro já existir, a informação deste campo irá indicar se o valor que está sendo recebido no arquivo será atualizado no registro encontrado. Este parâmetro não poderá ser verdadeiro caso o parâmetro “É chave” também seja.

Os modelos de transferência possuem ainda a possibilidade de configurar relacionamentos entre as composições através do ícone  (denominado Registro pai). Cada composição poderá ter apenas um registro pai, sendo necessário informar qual será o identificador da composição da qual ela é dependente. Para complementar o cadastramento do relacionamento será necessário acionar o ícone  de modo a incluir os campos de origem e destino.

Tendo os modelos necessários devidamente configurados é possível executar as três funções de transferência de dados disponíveis na ferramenta EDX:

- Migração: este processo tem como função transferir os dados de um banco para outro sem a necessidade de arquivos adicionais, ou seja, a ferramenta estabelece uma conexão com o banco origem e outra com o banco destino simultaneamente. Estabelecidas as conexões, a ferramenta iniciará uma transferência direta de dados entre os bancos.
- Exportação: a finalidade deste processo é gerar um documento XML com os dados extraídos do banco de origem, formatando-os conforme o

modelo de exportação da ferramenta.

- Importação: tendo um documento XML com dados formatados conforme as regras da ferramenta, será possível importar novos registros ou atualizar dados já existentes por meio deste processo.

A Figura 19 mostra a tela onde podem ser executados os processos relacionados acima.

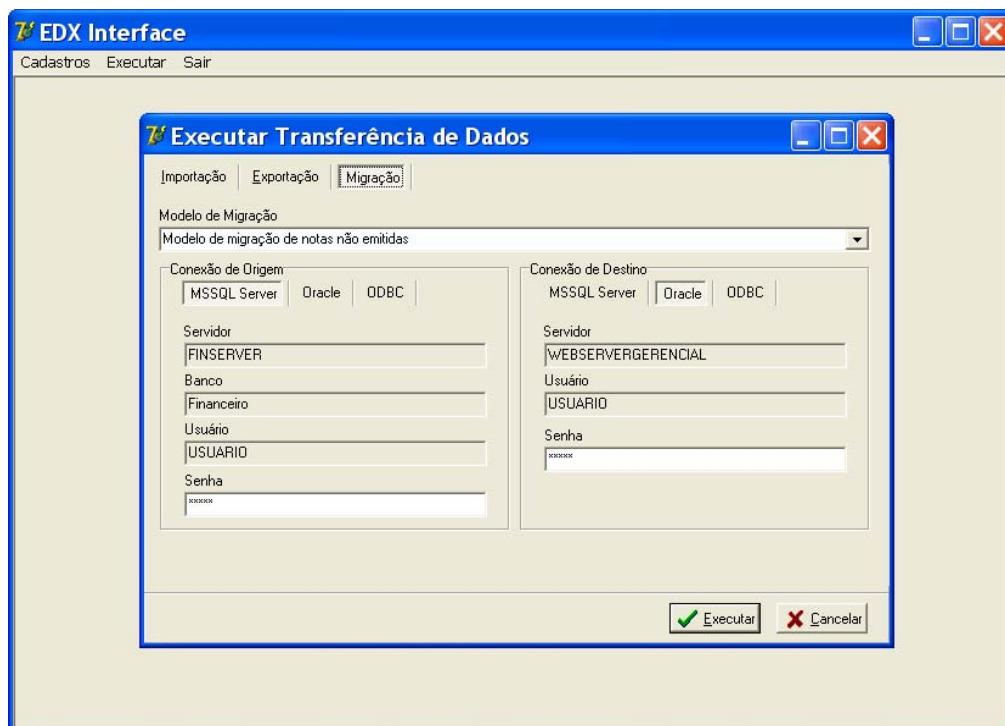


Figura 19: Tela de execução dos processos de transferência de dados.

Para execução do processo de importação será necessário informar o arquivo XML (inclusive com o caminho onde este se encontra) onde estão os dados a serem importados. Em relação ao processo de exportação será necessário escolher um caminho e um nome para o arquivo XML com os dados da exportação. No processo de migração será preciso escolher o modelo de migração.

Em todos os processos será necessário informar a senha de acesso para efetuar a conexão com o banco de dados. No caso da migração serão solicitadas as senhas para os bancos de origem e de destino.

3.4 Implementação

A escolha da linguagem XML para o desenvolvimento do projeto se deu devido à sua flexibilidade na criação de documentos capazes de armazenar dados de forma estruturada e hierárquica que qualquer aplicação pode entender, seja usando um *parser* ou lendo como um simples arquivo texto. Outra grande vantagem é o uso de DTD's para a validação e padronização do documento, sem a necessidade de se desenvolver um programa específico para verificar a consistência de sua estrutura e dos dados contidos no arquivo.

Considerando que a plataforma Windows foi escolhida para ser o ambiente do projeto, no decorrer deste buscou-se uma tecnologia que pudesse ser compartilhada entre uma aplicação local e uma aplicação para a Web. Esta tecnologia foi utilizada na implementação do componente responsável por executar os processos de migração dos dados de um banco para outro e pela exportação e importação dos dados utilizando arquivos XML. A tecnologia que se adaptou a essa proposta foi a das DLL's *ActiveX*.

Para o desenvolvimento da ferramenta EDX foi necessário utilizar uma ferramenta de programação que pudesse trabalhar com XML, *ActiveX*, ODBC e ADO. A ferramenta Delphi foi utilizada para o desenvolvimento da ferramenta por ter suporte a todas as tecnologias requisitadas para este projeto.

A ferramenta EDX está dividida em duas partes: a *interface* visual EDX Interface (que vem a ser uma interface gráfica de usuário), desenvolvida em Delphi, e o componente EDX Component, também desenvolvido em Delphi utilizando a tecnologia *ActiveX*. As classes que representam as partes da ferramenta podem ser encontradas nos apêndices. No diagrama de classes serão exibidos e descritos os métodos de cada classe que compõe a ferramenta EDX. Também podem ser encontrados nos apêndices os diagramas de *use-case* referentes à modelagem da ferramenta EDX, seus componentes e processos. O *use-case* Efetuar cadastros tem a finalidade de apresentar como se darão os processos de cadastramento dos

parâmetros de configuração e modelos de transferência. O *use-case* Executar transferência de dados tem como objetivo descrever como serão os processos de transferência de dados.

3.4.1 EDX Interface

A *interface* EDX Interface é a parte visível para o usuário, onde será permitido cadastrar os modelos necessários ao funcionamento da ferramenta EDX através de telas de cadastro. Também é possível executar pela *interface* os processos de transferência fazendo chamadas aos processos do componente EDX Component.

Existem três documentos XML que armazenam as informações das regras de troca de dados:

- CfgModeloConexao.xml: este documento é responsável por armazenar as informações que permitirão a conexão com os bancos de dados.
- CfgTabelaConversao.xml: este documento contém as informações referentes às tabelas de conversão utilizadas nos processos de migração, importação e exportação quando for necessário converter o valor recebido (seja diretamente do banco de dados, seja de um arquivo XML) em outro valor antes de enviá-lo ao destino.
- CfgMigracao.xml: as informações deste documento são referentes ao processo de transferência direta entre dois bancos.
- CfgExportacao.xml: este documento possui as informações utilizadas para geração de arquivos com dados a serem importados por outros sistemas.
- CfgImportacao.xml: neste documento são armazenados os modelos utilizados para importar dados de um arquivo XML formatado conforme as regras da ferramenta EDX.

Como forma de garantir a integridade da estrutura dos arquivos XML utilizados pela ferramenta, foram elaborados DTD's contendo a estrutura de cada arquivo XML tratado pela ferramenta EDX:

- CfgModeloConexao.dtd: responsável por validar o documento CfgModeloConexao.xml.
- CfgTabelaConversao: responsável por validade o documento CfgTabelaConversao.xml.
- CfgMigração.dtd: responsável por validar o documento CfgMigracao.xml.
- CfgExportacao.dtd: responsável pela validação do documento CfgExportacao.xml.
- CfgImportacao.dtd: responsável pela validação do documento CfgImportacao.xml.
- Intercambio.dtd: este documento possui as regras de validação tanto do arquivo XML gerado com os dados de exportação, quanto para validação dos arquivos com dados para serem importados pela ferramenta.

A estrutura dos DTD's referidos acima podem ser encontradas nos apêndices. Fazendo a análise dos DTD's será possível visualizar como serão armazenadas as informações nos documentos XML correspondentes.

As senhas dos bancos não serão armazenadas dentro do arquivo de configuração dos bancos, portanto, sempre que for necessária a conexão a algum banco de dados, seja para manutenção dos modelos, seja pela execução dos processos, será solicitada a senha do banco.

3.4.2 EDX Component

O EDX Component, em virtude da tecnologia empregada em seu desenvolvimento (*ActiveX*), pode ser acionado tanto pela EDX Interface quanto por qualquer outro sistema que suporte tal tecnologia.

A *interface* EDX Interface pode executar qualquer método do componente EDX Component. Após a conclusão do processo, a *interface* retornará ao processo que a acionou um log com as operações realizadas, sejam elas concluídas com sucesso ou não.

Em relação aos processos de importação e migração a ferramenta EDX estabelece que seja aberta uma transação no banco de dados para cada conjunto de 100 registros transferidos, sendo encerrada quando o último destes for transferido. No caso de uma composição ser dependente (relacionada através da estrutura de registro pai) de outra, esta será integrante do registro da composição de mais alto nível, ou seja, uma composição que não é dependente de nenhuma outra composição.

Como forma de padronizar e evitar inconsistências de dados foi estabelecido que o tipo de dados Real deve estar formatado no padrão americano sem separador de milhar, ou seja, o separador decimal será o ponto final (.). Para os campos com formato no tipo Data o seu conteúdo será formatado e aceito sem separadores e no formato AAAAMMDD, onde AAAA= Ano, MM = Mês e DD = Dia.

No caso dos campos do tipo *String* e Memo, caso estes possuam algum caractere reservado da linguagem XML, os mesmos serão convertidos durante o processo de exportação. No caso do processo de importação não serão aceitos arquivos XML contendo caracteres reservados da linguagem. Para que estes tipos de campos possam receber caracteres específicos da língua portuguesa será utilizada a codificação ISSO-8859-1.

3.4.3 Modelagem

Para modelagem da ferramenta foi utilizada a técnica UML. Aqui neste capítulo estão destacadas as interações entre as classes que compõem a ferramenta EDX através de diagramas de seqüência.

O papel do usuário dentro do contexto dos diagramas a seguir se limita exclusivamente a cadastrar os modelos e solicitar a execução dos processos. O usuário também poderá decidir sobre as ações a tomar em virtude das mensagens de retorno da EDX Interface.

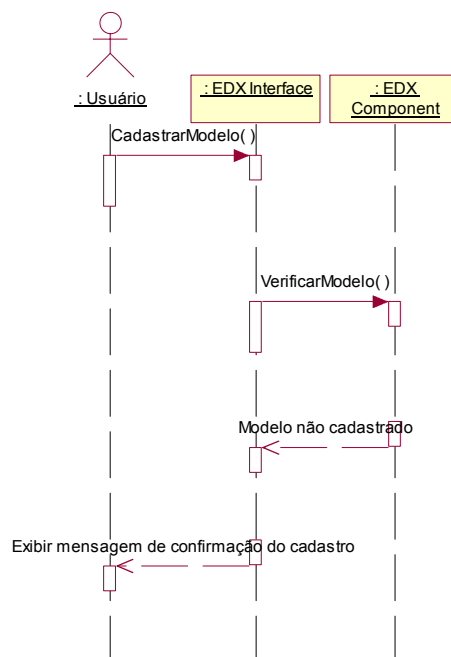


Figura 20: Diagrama de seqüência dos processos de cadastro dos modelos.

A Figura 20 exibe a seqüência dos processos de cadastramento dos modelos utilizados pela ferramenta. Ao solicitar o cadastro de um novo modelo, a EDX Interface irá solicitar ao EDX Component que verifique se já existe um modelo com o mesmo identificador, caso não exista o componente retornará esta situação à EDX Interface que realizará a gravação do modelo e exibirá a confirmação do cadastro ao usuário. No caso de já haver um modelo com o mesmo identificador do modelo que se deseja incluir, o componente irá retornar esta situação e a EDX Interface solicitará

ao usuário que modifique o identificador. Esta seqüência se repetirá até que o componente retorne uma situação de não cadastrado ou o usuário cancele a gravação do modelo.

Para todos os modelos utilizados pela ferramenta EDX o processo de cadastramento seguirá a seqüência apresentada na Figura 20.

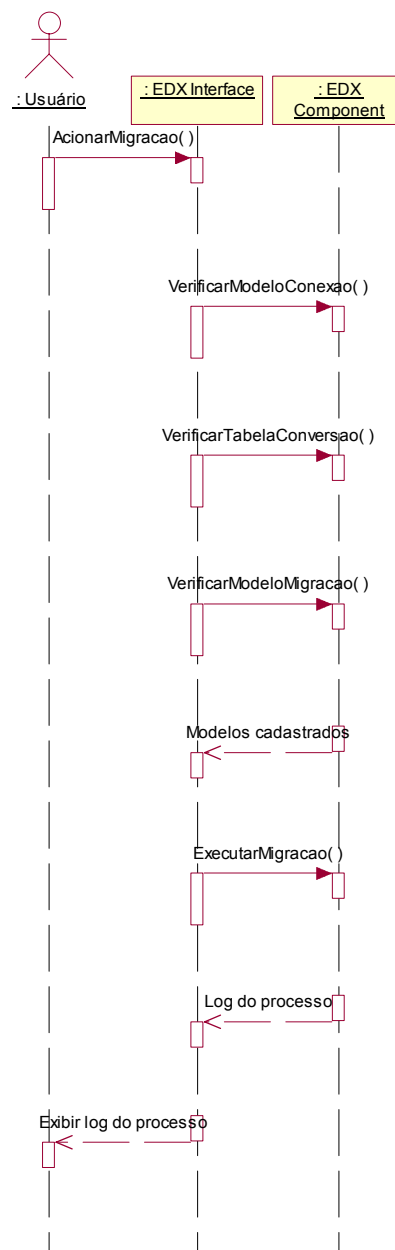


Figura 21: Diagrama de seqüência do processo de migração.

Na Figura 21 pode-se visualizar a seqüência de passos do processo de migração de dados.

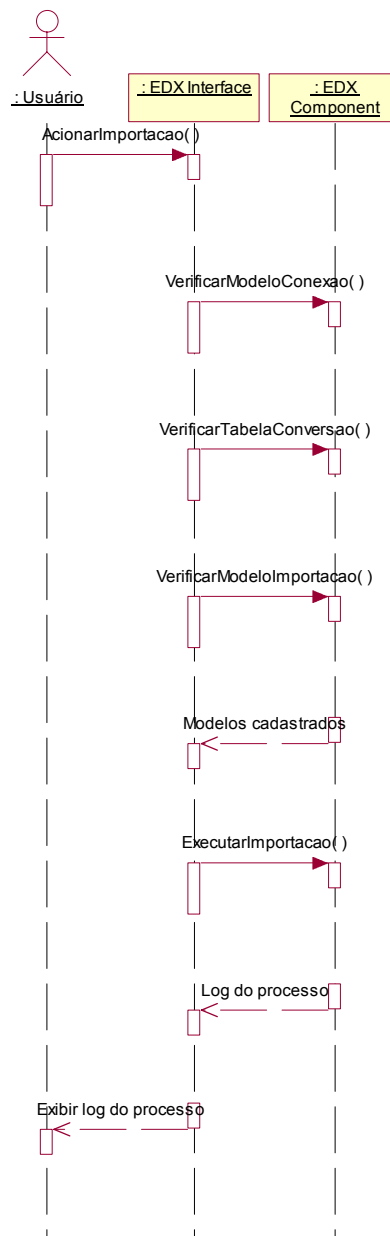


Figura 22: Diagrama de seqüência do processo de importação.

Na figura 22 pode-se perceber que para o processo de importação de dados há o mesmo relacionamento entre as duas classes e o usuário que existente no diagrama do processo de migração.

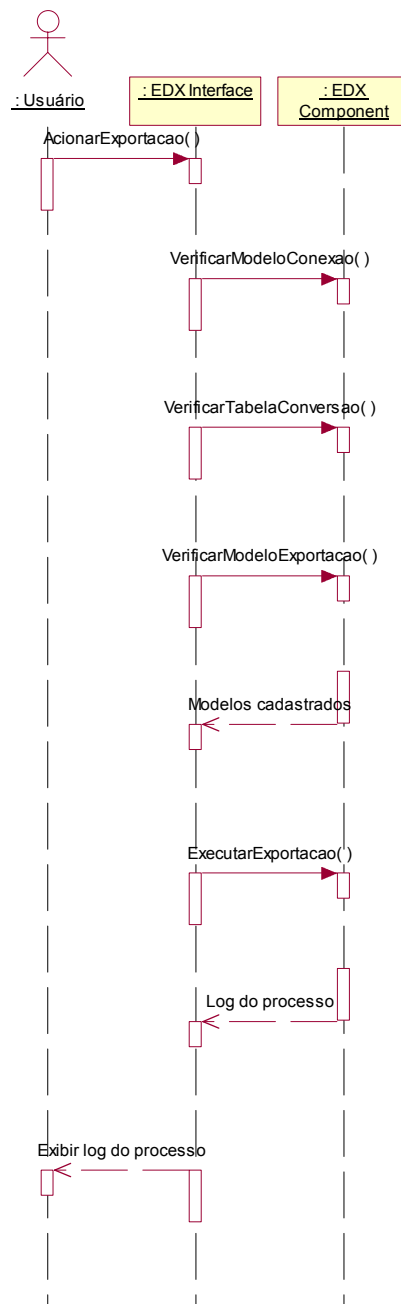


Figura 23: Diagrama de seqüência do processo de exportação.

Finalizando os diagramas de seqüência com a Figura 23, pode-se verificar claramente que os métodos chamados para verificar os cadastros dos modelos de conexão e das tabelas de conversão, são referenciados em todos os processos de transferência de dados. Isto significa que para executar qualquer processo de transferência será necessário cadastrar todos os modelos referentes ao processo

que se deseja executar.

Os diagramas representados nas Figuras 21, 22 e 23 demonstram processos de transferência executados com sucesso. Todos os modelos foram verificados e o EDX Component retornou que todos estavam cadastrados. A partir disso a EDX Interface solicitou a execução do processo de transferência, obtendo um log com as ocorrências do processo que foi concluído com sucesso.

No entanto, podem ocorrer erros durante um processo de transferência, que podem ocorrer em relação aos modelos ou em relação aos dados de transferência. Por exemplo, em um processo de exportação, se a tabela de conversão referenciada no modelo de exportação não estiver cadastrada nos modelos de tabela de conversão, o EDX Component reportará o erro e a EDX Interface exibirá ao usuário que o modelo não se encontra cadastrado. Em virtude deste fato o processo de exportação não terá continuidade. Caberá ao usuário alterar o modelo de exportação ou cadastrar a referida tabela de conversão antes de executar novamente o processo.

Em relação aos dados, pode acontecer uma queda na conexão com o SGBD (por ter ocorrido um problema na rede local, por exemplo), ou então durante a inclusão de um registro em um processo de importação, uma determinada coluna da tabela (sendo esta coluna uma chave estrangeira) foi preenchida com um valor que não se encontra cadastrado na tabela com a qual esta coluna se relaciona, ocasionando uma falha de integridade referencial. Em casos como estes os registros não serão inseridos, bem como seus registros dependentes, e os erros serão retornados à EDX Interface e esta os exibirá ao usuário para que este faça as devidas correções antes de executar novamente o processo.

CAPÍTULO IV – CONCLUSÃO E TRABALHOS FUTUROS

4.1 Conclusão e Trabalhos Futuros

Este projeto mostra uma das muitas possibilidades de utilização da linguagem XML. O enfoque dado aqui neste trabalho representa apenas uma pequena parte do que se pode conseguir quando a linguagem XML for bem trabalhada.

Utilizar o XML como base para transferência de dados se mostrou bastante eficiente e trouxe uma grande flexibilidade na execução dos processos de transferência de dados. Por ser permitida uma validação automática de sua estrutura, utilizando-se de DTD's, os programas de leitura podem ser mais simplificados, pois não terão que verificar, por exemplo, se o arquivo possui a quantidade de caracteres permitidos para uma linha. Outra funcionalidade extremamente interessante de XML para o desenvolvimento de aplicações de transferência de dados está no fato de poder ser dado um nome para cada informação. Isto evita os eventuais erros em arquivos de formatação fixa (como os arquivos CNAB) onde uma linha pode ser gerada com uma determinada coluna deslocada para direita, ocasionando um erro de execução, ou o que pode ser mais grave, a leitura de uma informação incorreta.

Utilizar a ferramenta EDX evita que a cada necessidade de uma transferência de dados seja necessário que se faça um programa específico ou alteração em um programa existente, pois bastará alterar um dos modelos de transferência ou criação de um outro modelo para atender à nova necessidade.

Sendo o EDX Component desenvolvido em *ActiveX* será possível que qualquer aplicação desenvolvida em uma linguagem que suporte esta tecnologia possa acionar a execução dos processos de transferência. Isto se torna interessante

na medida em que será possível agendar a execução dos processos, como por exemplo: ao final do dia o sistema de estoque pode atualizar o banco que permite acesso via Web com os novos dados de preço e quantidades em estoque dos produtos.

No início da elaboração deste trabalho não se tinha idéia do quanto a escolha da linguagem XML fosse contribuir para o enriquecimento da ferramenta EDX. Como dito anteriormente, a possibilidade de se utilizar DTD's e nomear as informações tornou o programa mais simples de ser implementado, fazendo deste um desenvolvimento mais rápido e de melhor qualidade.

Todavia, durante a finalização do projeto foram visualizadas algumas melhorias que poderiam ser adicionadas à ferramenta EDX:

- Permitir que a origem dos dados seja um comando SQL (*Structure Query Language*). Esta implementação possibilitará que os dados de origem possam ser uma junção de tabelas, com funções, agrupamentos e demais funcionalidades que a linguagem SQL pode oferecer. Podem existir casos onde será necessário gerar arquivos com a média de vendas dos últimos meses, ou fazer uma migração de dados onde o destino deva receber, por exemplo, as maiores compras de cada mês. O uso de comandos SQL na configuração da origem dos dados visa estender a aplicação da ferramenta, pois além de permitir a utilização das funções de agregação (com Min, Max e Avg), será possível selecionar visões dos dados antes de realizar os processos de transferência;
- Importação de arquivos texto contendo dados em um formato fixo. Algumas vezes não é possível uma conexão direta com o banco de dados de origem (como exemplos têm-se sistemas antigos baseados em *mainframes*). Uma prática comum nestas situações é a geração dos dados de origem em arquivos texto formatados que possam ser lidos por um programa especialmente feito para a importação de tais informações. Criando na ferramenta a possibilidade de importar tais arquivos seria evitada a necessidade da criação de programas específicos para a leitura

dos mesmos. Outro caso que torna interessante a adição desta funcionalidade ocorre, principalmente, no relacionamento com os bancos (instituições financeiras) e outras empresas onde a troca de dados é feita através de arquivos texto de formatação fixa, sendo que não há a possibilidade de que sejam trocados arquivos XML nos formatos suportados pela ferramenta EDX.

Estas novas funcionalidades poderão, futuramente, ser implementadas na ferramenta modo a deixá-la mais abrangente e funcional.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDERSON, R. *et al. Professional XML*. Rio de Janeiro: Ciência Moderna, 2001.

BROWN, S. *Visual Basic 6: bíblia do programador*. Tradução de Lars Gustav Erik Unonius. São Paulo: Berkeley Brasil, 1999.

CANTÙ, M. *Dominado o Delphi: a bíblia*. Tradução de Kátia Aparecida Roque e revisão técnica de Álvaro Rodrigues Antunes. São Paulo: Pearson Education do Brasil, 2003.

COLCHER, R; VALLE, A. *Guia de EDI e Comércio Eletrônico*. Rio de Janeiro: Simpro Brasil, 2000.

CORREIA, M. *EDI-MHS: a comunicação empresarial global*. São Paulo: Érica, 1991.

DÉCIO, O. C. *XML: guia de consulta rápida*. São Paulo: Novatec, 2000.

Deitel, H.M. *et al. XML: como programar*. Tradução de Luiz Augusto Salgado e Edson Furmankiewicz. Porto Alegre: Bookman, 2003.

FEBRABAN. *Intercâmbio de Informações entre Bancos e Empresas: padrão FEBRABAN 240 posições*. Versão: 6.0. 2003.

ITAU. *Cobrança Bancária: intercâmbio eletrônico de arquivos*. 2000.

KORTH, H. F.; SILBERSCHATZ, A. *Sistema de Banco de Dados*. Tradução de Maurício Heihachiro Abe e revisão técnica de Sílvio Carmo Palmieri. 2ª ed. São Paulo: Makron Books, 1993.

MCGRATH, S. *XML: aplicações práticas*. Tradução de Vitor Hugo da Paixão Alves. Rio de Janeiro: Campus, 1999.

MICROSOFT. *Visual Basic Guia de Ferramentas Componentes*: Sistema de programação para windows. 1997.

O'BRIEN, J. A. *Sistemas de Informação e as Decisões Gerenciais na Era da Internet*. Tradução de Cid Knipel Moreira. São Paulo: Saraiva, 2003.

OGBUJI, U. *XML: The Future of EDI?* <http://xml.coverpages.org/ogbuji-swol-02-xmledi.html>. Recolhido em 01/2004.

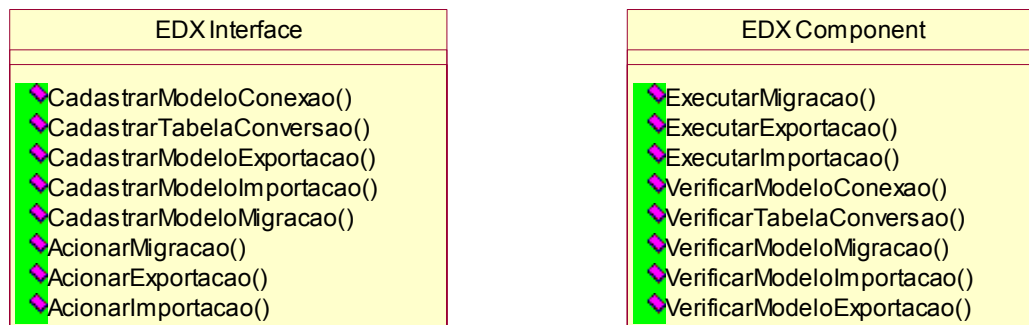
PITTS-MOULTIS, N.; KIRK, C. *XML Black Book*. Tradução de Ariovaldo Griesi e revisão técnica de Álvaro Rodrigues Antunes. São Paulo: Makron Books, 2000.

W3C. *Extensible Markup Language (XML)*. <http://www.w3.org/XML>. Recolhido em 03/2004.

W3C. *HiperText Markup Language (HTML)*. <http://www.w3.org/MarkUp/>. Recolhido em 03/2004.

APÊNDICES

Apêndice A – Diagrama de Classes



Métodos da classe EDXInterface:

- CadastrarModeloConexao: Método responsável pelo cadastramento dos modelos de conexão.
- CadastrarTabelaConvesão: Método responsável pelo cadastramento das tabelas de conversão utilizadas nos modelos de importação, migração e exportação.
- CadastrarModeloExportacao: Método responsável pelo cadastramento dos modelos de exportação da ferramenta EDX.
- CadastrarModeloImportacao: Método responsável pelo cadastramento dos modelos de importação.
- CadastrarModeloMigracao: Método responsável pelo cadastramento dos modelos de migração.
- AcionarMigracao: Este método faz uma chamada ao método ExecutarMigracao do componente EDX Component.
- AcionarImportacao: Este método faz uma chamada ao método ExecutarImportacao do componente EDX Component.
- AcionarExportacao: Este método faz uma chamada ao método ExecutarExportacao do componente EDX Component.

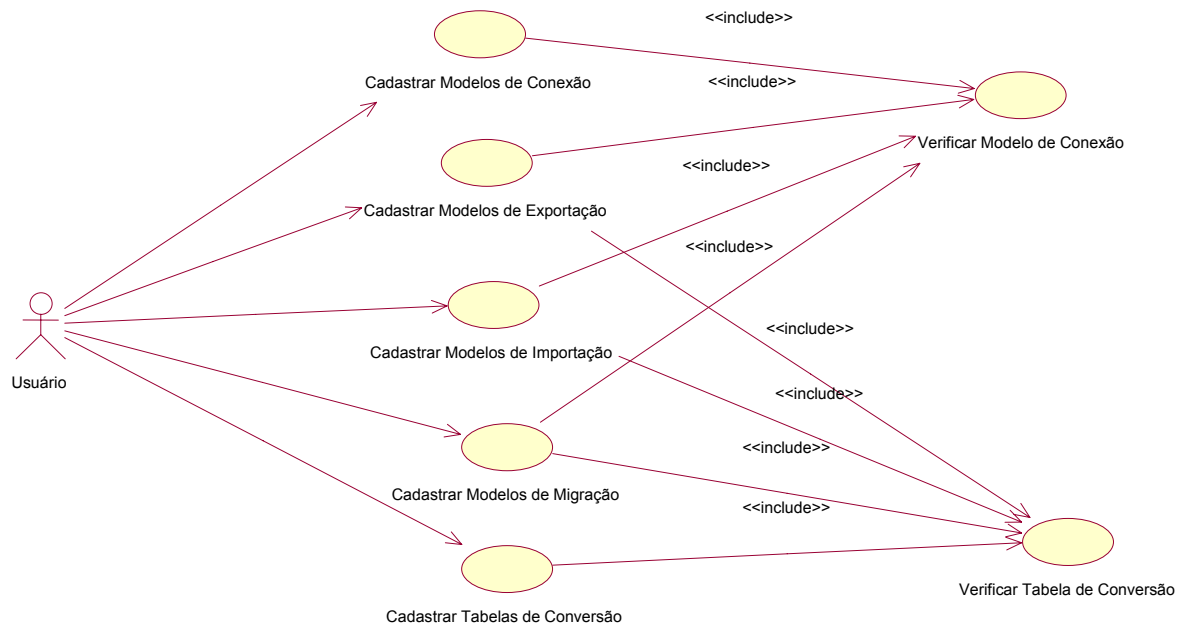
Métodos da classe EDXComponent:

- VerificarModeloConexao: Método que verifica se o modelo de conexão existe no arquivo CfgModeloConexao.xml.

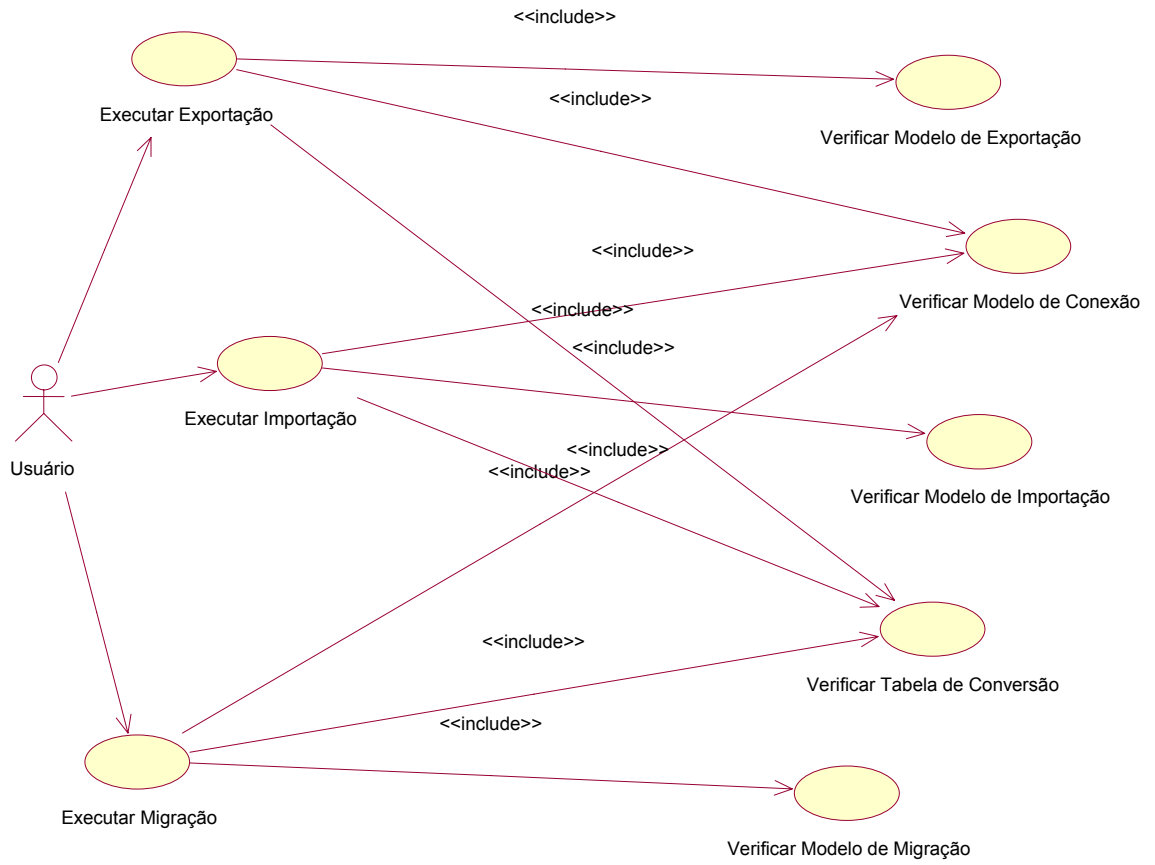
- VerificarTabelaConversao: Método responsável por verificar se a tabela de conversão existe no arquivo CfgTabelaConversao.xml.
- VerificarModeloImportacao: Método que verifica se o modelo de importação está cadastrado no arquivo CfgModeloImportacao.xml.
- VerificarModeloExportacao: Método que verifica se o modelo de exportação está presente no arquivo CfgModeloExportacao.xml.
- VerificarModeloMigracao: Método responsável por verificar se o modelo de migração está cadastrado no arquivo CfgModeloMigracao.xml.
- ExecutarMigracao: Método que executa a migração de dados com base nos parâmetros de migração.
- ExecutarImportação: Método que executa a migração de dados com base nos parâmetros de importação.
- ExecutarExportação: Método que executa a exportação de dados com base nas parâmetros de exportação.

Apêndice B – Diagramas de Use-case

Efetuar cadastros



Executar transferências de dados



Apêndice C – Documentos DTD

CfgConexao.dtd

```

<!ELEMENT Conexoes (Conexao*) >
<!ELEMENT Conexao (MSSQLServer|Oracle|DSN ) >
  <!ATTLIST Conexao Identificador CDATA #REQUIRED >
<!ELEMENT MSSQLServer (Servidor ,Banco ,Usuario ) >
<!ELEMENT Servidor (#PCDATA) >
<!ELEMENT Banco (#PCDATA) >
<!ELEMENT Usuario (#PCDATA) >
<!ELEMENT Oracle (Servidor ,Usuario ) >
<!ELEMENT DSN (#PCDATA) >

```

CfgTabelaConvesao.dtd

```

<!ELEMENT TabelasConversao (Tabela+) >
<!ELEMENT Tabela (Item+) >
  <!ATTLIST Tabela Identificador CDATA #REQUIRED >
<!ELEMENT Item (Conversao+) >
  <!ATTLIST Item Identificador CDATA #REQUIRED >
<!ELEMENT Conversao (ValorOrigem, ValorDestino) >
<!ELEMENT ValorOrigem (#PCDATA) >
<!ELEMENT ValorDestino (#PCDATA) >

```

CfgMigracao.dtd

```

<!ELEMENT Modelos (Modelo+ ) >
<!ELEMENT Modelo (CfgConexoes ,TabelaConversao ,Composicoes ) >
  <!ATTLIST Modelo Identificador CDATA #REQUIRED >
<!ELEMENT CfgConexoes (IdentificadorOrigem ,IdentificadorDestino ) >
<!ELEMENT IdentificadorOrigem (#PCDATA) >
<!ELEMENT IdentificadorDestino (#PCDATA) >
<!ELEMENT TabelaConversao (#PCDATA) >
<!ELEMENT Composicoes (Composicao+) >
<!ELEMENT Composicao (TabelaOrigem ,TabelaDestino ,Campos ,RegistroPai ) >
  <!ATTLIST Composicao Identificador CDATA #REQUIRED >
<!ELEMENT TabelaOrigem (Nome ,Filtro ) >
<!ELEMENT Nome (#PCDATA) >
<!ELEMENT Filtro (#PCDATA) >
<!ELEMENT TabelaDestino (Nome ) >
<!ELEMENT Campos (Campo+) >
<!ELEMENT Campo (Origem ,Destino ,Chave ,PermiteAtualizacao ) >
<!ELEMENT Origem (Tabela|Conversao|Fixo) >
<!ELEMENT Tabela (Nome ) >
<!ELEMENT Nome (#PCDATA) >
  <!ATTLIST Nome Tipo CDATA #REQUIRED >

```

```

<!ELEMENT Conversao (Nome ) >
<!ELEMENT Nome (#PCDATA) >
  <!ATTLIST Nome Tipo CDATA #REQUIRED >
<!ELEMENT Fixo (#PCDATA) >
<!ELEMENT Destino (Tabela|Conversao|Fixo) >
<!ELEMENT Tabela (Nome ) >
<!ELEMENT Nome (#PCDATA) >
  <!ATTLIST Nome Tipo CDATA #REQUIRED >
<!ELEMENT Conversao (Nome ) >
<!ELEMENT Nome (#PCDATA) >
  <!ATTLIST Nome Tipo CDATA #REQUIRED >
<!ELEMENT Fixo (#PCDATA) >
<!ELEMENT Chave (#PCDATA) >
<!ELEMENT PermiteAtualizacao (#PCDATA) >
<!ELEMENT RegistroPai (Chave+ ) >
  <!ATTLIST RegistroPai Identificador CDATA #REQUIRED >
<!ELEMENT Chave (CampoTabelaPai ,CampoTabelaFilha ) >
<!ELEMENT CampoTabelaPai (#PCDATA) >
<!ELEMENT CampoTabelaFilha (#PCDATA) >

```

CfgExportacao.dtd

```

<!ELEMENT Modelos (Modelo ) >
<!ELEMENT Modelo (CodigoIntercambio ,CfgConexao ,TabelaConversao ,Composicoes ) >
  <!ATTLIST Modelo Identificador CDATA #REQUIRED >
<!ELEMENT CodigoIntercambio (#PCDATA) >
<!ELEMENT CfgConexao (#PCDATA) >
<!ELEMENT TabelaConversao (#PCDATA) >
<!ELEMENT Composicoes (Composicao+) >
<!ELEMENT Composicao (TabelaOrigem ,Filtro ,Campos ,RegistroPai ) >
  <!ATTLIST Composicao Identificador CDATA #REQUIRED >
<!ELEMENT TabelaOrigem (#PCDATA) >
<!ELEMENT Filtro (#PCDATA) >
<!ELEMENT Campos (Campo+) >
<!ELEMENT Campo (Tabela|Conversao|Fixo ) >
  <!ATTLIST Campo Identificador CDATA #REQUIRED >
<!ELEMENT Tabela (Nome ) >
<!ELEMENT Nome (#PCDATA) >
  <!ATTLIST Nome Tipo CDATA #REQUIRED >
<!ELEMENT Conversao (Nome ) >
<!ELEMENT Nome (#PCDATA) >
  <!ATTLIST Nome Tipo CDATA #REQUIRED >
<!ELEMENT Fixo (#PCDATA) >
<!ELEMENT RegistroPai (Chave+ ) >
  <!ATTLIST RegistroPai Identificador CDATA #REQUIRED >
<!ELEMENT Chave (CampoTabelaPai ,CampoTabelaFilha ) >
<!ELEMENT CampoTabelaPai (#PCDATA) >
<!ELEMENT CampoTabelaFilha (#PCDATA) >

```

CfgImportacao.dtd

```

<!ELEMENT Modelos (Modelo ) >
<!ELEMENT Modelo (Descricao ,CfgConexao ,TabelaConversao ,Composicoes ) >
  <!ATTLIST Modelo CodigoIntercambio CDATA #REQUIRED >
<!ELEMENT Descricao (#PCDATA) >

```

```

<!ELEMENT CfgConexao (#PCDATA) >
<!ELEMENT TabelaConversao (#PCDATA) >
<!ELEMENT Composicoes (Composiçã+) >
<!ELEMENT Composicao (TabelaDestino ,PermiteAtualizacao ,Campos ,RegistroPai ) >
  <!ATTLIST Composicao Identificador CDATA #REQUIRED >
<!ELEMENT TabelaDestino (#PCDATA) >
<!ELEMENT PermiteAtualizacao (#PCDATA) >
<!ELEMENT Campos (Campo+) >
<!ELEMENT Campo (Tabela|Conversao|Fixo ,Chave ,PermiteAtualizacao ) >
  <!ATTLIST Campo Identificador CDATA #REQUIRED >
<!ELEMENT Tabela (Nome ) >
<!ELEMENT Nome (#PCDATA) >
  <!ATTLIST Nome Tipo CDATA #REQUIRED >
<!ELEMENT Conversao (Nome ) >
<!ELEMENT Nome (#PCDATA) >
  <!ATTLIST Nome Tipo CDATA #REQUIRED >
<!ELEMENT Fixo (Nome ,Valor ) >
<!ELEMENT Valor (#PCDATA) >
<!ELEMENT Chave (#PCDATA) >
<!ELEMENT RegistroPai (Chave+) >
  <!ATTLIST RegistroPai Identificador CDATA #REQUIRED >
<!ELEMENT CampoTabelaPai (#PCDATA) >
<!ELEMENT CampoTabelaFilha (#PCDATA) >

```

Intercambio.dtd

```

<!ELEMENT Intercambio (CodigoIntercambio ,Composiçã+) >
<!ELEMENT CodigoIntercambio (#PCDATA) >
<!ELEMENT Composicao (Campo+) >
  <!ATTLIST Composicao Identificador CDATA #REQUIRED >
<!ELEMENT Campo (#PCDATA) >
  <!ATTLIST Campo Identificador CDATA #REQUIRED >

```