Aproximando XML ao Mundo Relacional
Um Framework para Consultas Genéricas a Banco de Dados

Fernanda Emanuella Silveira
Sérgio Carlos Castelani Júnior

6 Fevereiro 2003

Resumo

Uma das principais etapas no desenvolvimento de uma aplicação para comunicação de dados é o projeto do protocolo usado na transferência dos dados. Assim, cada aplicação de um lado da comunicação deverá receber e enviar os dados da maneira como foi especificado a ela para que possa haver a troca efetiva de informação. Porém, se cada aplicação desenvolvida implementar o seu próprio protocolo, a intercomunicação entre aplicações diferentes que desejarem trocar informações será dificultada. O que se propõe atualmente é usar uma linguagem padronizada para a troca de informação chamada XML (Extended Markup Language) . Ela permite que sejam enviados não apenas os dados, mas também o significado deles. Todas as aplicações poderão se beneficiar com esta idéia uma vez que precisarão se preocupar em entender somente um tipo padrão de linguagem. Como, normalmente, todas as aplicações que trocam informações utilizam um banco de dados central, torna-se imprescindível que os servidores de banco de dados saibam dispor os seus dados em XML. Propõe-se, portanto, um framework capaz de fornecer o conteúdo de qualquer banco de dados, em qualquer formatação, XML ou não, para as aplicações.

Abstract

One of the most important stages in the application for data communication development is the project of the protocol used to transfer the information. Each application at a side of the communication must send and receive data in the way they were specified to realize the effective information exchange. However if each developed application implements its own protocol the intercommunication between different applications that wish to exchange information will be difficulted. What is suggested nowadays is to use a standardized language to exchange information called XML (Extended Markup Language). It permits sending both data and their meaning. All applications can take advantage of this idea once they will be concerned about understanding only one type of language. Since most applications that work with information uses central data bases, it's important that the database servers know how to interpret a XML request. A framework is proposed to provide the content of any database, in any format, XML or not, to the application.

Agradecimentos

"Gostaria de agradecer todo o apoio que recebi nestes últimos meses por parte de minha família, amigos, professores, mas principalmente de minha companheira de trabalho Fernanda Emanuela Silveira e de minha namorada Fernanda Amaral Silveira (incrível, não!), que sempre levantaram a minha moral nos momentos de desânimo e desistência. Um forte abraço a todos!"

Sérgio C Castelani Jr.

"Agradeço imensamente às pessoas ao meu redor, a minha família, ao meu pai, minha mãe e minha irmã, ao meu amigo e colega de trabalho Sérgio Castelani Jr e principalmente a Deus por mais um obstáculo vencido."

Fernanda Emanuela Silveira.

"Agradecemos ao nosso orientador Professor Antônio Augusto Fröhlich por seu apoio intelectual, moral e por sua compreensão durante todo o processo de elaboração deste trabalho e aos membros da banca: Professor Aldo von Wangenheim e o Mestrando Marcelo Trierveiler Pereira."

Sérgio Carlos Castelani Jr. e Fernanda Emanuella Silveira.

Conteúdo

Capítulo 1

Introdução

1.1 Objetivos

1.1.1 Objetivo Geral

Criar um framework para tratar qualquer tipo de consulta a dados armazenados em um banco de dados.

1.1.2 Objetivos Específicos

Implementação de um servidor de banco de dados, usando o framework proposto, que ficará esperando uma consulta SQL de aplicações clientes através da interface e retornará um documento de acordo com a solicitação, contendo os dados requisitados.

Desenvolvimento de uma aplicação cliente que utilize os recursos do servidor a fim de validar este framework.

1.2 Metodologia

1.2.1 Linguagem de Programação

A linguagem escolhida para implementação do framework foi o Pascal devido a experiencias anteriores de programação e familiaridade com a mesma e o ambiente de desenvolvimento utilizado foi o Delphi 6.0 pelas facilidades oferecidas.

1.2.2 Pesquisa sobre o Estado da Arte

Para se obter um cenário atual e real do andamento e difusão de tecnologias relacionadas a XML, acesso a banco de dados, padronização na maneira de representar informação e temas associados fez-se uma intensa pesquisa.

Para tanto usou-se a Internet, livros da área, trabalhos e artigos publicados no meio acadêmico e comercial no intuito de utilizar e avaliar as ferramentas existentes e desenvolver um trabalho voltado a alguma necessidade do mercado atual.

1.3 Motivação

Atualmente muito se tem falado em padronizar a transmissão de informações, usando XML, mas na prática não há um uso em massa deste recurso que pode facilitar a comunicação de sistemas heterogêneos.

Homepages inteiras são reescritas quando é preciso mudar apenas o formato de sua apresentação, exigindo um trabalho dobrado de reformulação de conteúdo e de apresentação. Nessa troca, informações são perdidas, podendo ter até seu significado alterado.

1.4 Justificativa

Com o crescimento da disponibilidade e busca de informação na Internet e nas redes de computadores em geral, sobretudo do ponto de vista comercial, estão aparecendo cada vez mais aplicações que utilizam o padrão XML para armazenar, transferir ou manipular os dados.

Muitos esforços estão sendo feitos no sentido de armazenar dados em XML na forma de um banco de dados. Porém nenhum deles está preocupado com a abrangência das aplicações que utilizarão estes dados. Em todos os tipos de tratamento desenvolvidos, o banco de dados resultante trabalhará com documentos XML que seguem somente uma estrutura definida.

Isso significa que as aplicações que trabalharão com esse banco deverão utilizar documentos XML que seguem exatamente aquele tipo de estrutura.

Porém, em um ambiente comercial, onde existem vários tipos de softwares de vários fabricantes diferentes, ficaria praticamente impossível fazer com que todos os aplicativos que tratassem de um mesmo tipo de informação seguissem uma mesma estrutura de documento. Outro problema seria criar uma aplicação que coleta dados de vários bancos de dados diferentes. Esta aplicação teria que saber interpretar cada estrutura XML de cada banco.

Um exemplo prático:

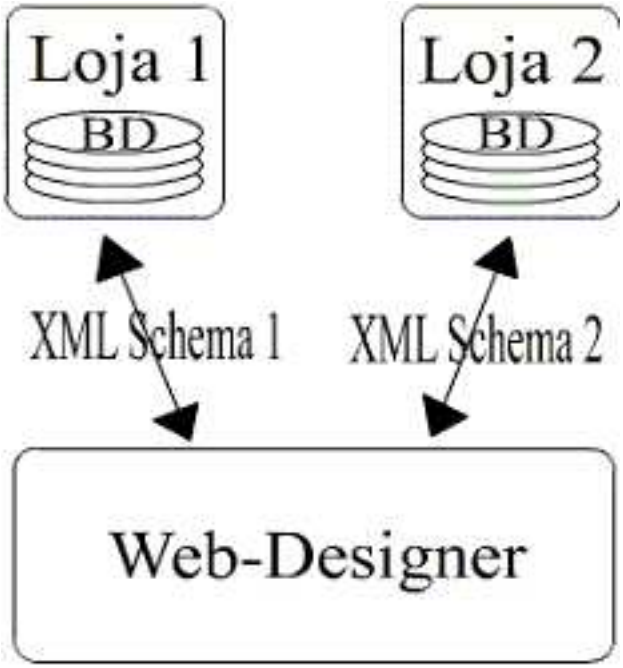

Figura 1.1: Aplicação de Banco de Dados com XML

Imagine uma empresa de desenvolvimento de websites que tenha duas lojas de grande porte distintas como cliente. Cada loja possui um banco de dados que relaciona, entre outras coisas, as peças em estoque. Agora digamos que a empresa de web-design deseja desenvolver um software que acesse o banco de dados de cada cliente e adquira informações sobre todas as peças em estoque para que as suas respectivas páginas fiquem sempre atualizadas. Se cada cliente utiliza um banco de dados diferente e, além disso, cada um utiliza a sua estrutura XML para os softwares internos, o web-designer teria que escrever o seu programa de modo que pudesse interpretar os documentos de cada cliente. Agora imagine se esta empresa trabalhasse com 10, 20 ou 30 clientes!

O que se pretende com este trabalho, é desenvolver um modo de acabar com esta limitação dos bancos de dados que trabalham com XML. Estes deverão ser capazes de disponibilizar os seus dados seguindo qualquer estrutura XML e não ficarem amarrados a uma estrutura só. Assim o problema do web-designer poderia ser resolvido pedindo para que os clientes mandassem o seu conteúdo seguindo um mesmo formato, definido por ele mesmo na hora da requisição.

Capítulo 2

Conceituação Teórica

2.1 XML

O desenvolvimento do XML (Extended Markup Language) foi baseado no SGML (Standard Generalized Markup Language) que permite a definição de estruturas para documentos e no HTML (Hypertext Markup Language) que é uma aplicação de SGML que possui um conjunto fixo de marcações ou tags, como são conhecidas.

XML é uma linguagem semântica que permite a apresentação, a troca e a gerência de dados juntamente com o seu significado. Sintaticamente, documentos XML se parecem a documentos HTML.

Um documento XML bem formado, isto é, que esteja em conformidade com a sintaxe XML, começa com um prólogo e contém exatamente um elemento, também chamado de elemento raiz do documento. Adicionalmente podem ser incluídos comentários e instruções de processamento.

Exemplo de um documento XML:

```
<?xml version="1.0" standalone="yes" encoding="UTF-8"?>

<!-- Este é um exemplo de um catálogo de biblioteca. -->

<BIBLIOTECA>
<LIVRO>
    <AUTOR id="aho"> Aho, A. V. </AUTOR>
    <AUTOR id="sethi"> Sethi, R. </AUTOR>
    <AUTOR id="ullman"> Ullman, J. D. </AUTOR>
    <TITULO> Compilers: Principles, Techniques, and Tools </TITULO>
    <EDITORA> Addison-Wesley </EDITORA>
    <ANO> 1985 </ANO>
```

```
</LIVRO>

   <LIVRO>
    <AUTOR id="tanenbaum"> Tanenbaum, A. S. </AUTOR>
    <TITULO> Computer Networks </TITULO>
    <EDITORA> Prentice-Hall </EDITORA>
    <ANO> 1996 </ANO>
</LIVRO>
</BIBLIOTECA>
```

Neste exemplo o prólogo

```
<?xml version="1.0" standalone="yes" encoding="UTF-8"?>
```

informa que o documento segue a versão 1.0 do XML, é stand-alone, ou seja, não está acompanhado de uma DTD, e utiliza a codificação UTF-8 (Unicode Transfromation Format 8-bit).

Se o documento estivesse acompanhado de uma DTD, a declaração da DTD também faria parte do prólogo. Tudo que está entre

```
<!-- e -->
```

é comentário. O elemento raiz é BIBLIOTECA e contém o elemento LIVRO, que por sua vez contém outros elementos como AUTOR, TITULO, EDITORA e ANO. Cada elemento pode ter um atributo associado, como id="aho", em AUTOR. O valor do atributo deve estar entre aspas.

2.2 DTD

DTD (Document Type Definition), que também é usada no SGML, define a estrutura dos documentos XML. Pode-se pensar em uma DTD como sendo uma gramática livre de contexto (GLC).

Com uma DTD é possível especificar o conjunto de tags, a ordem das tags e os atributos associados a cada uma delas. Um documento XML bem formado e que esteja em conformidade a sua DTD é chamado de válido.

Exemplo de uma DTD:

```
<!DOCTYPE biblioteca [
<!ELEMENT BIBLIOTECA (LIVRO+)>
<!ELEMENT LIVRO (AUTOR+, TITULO, EDITORA?, ANO?)>
<!ELEMENT AUTOR (#PCDATA)>
<!ATTLIST AUTOR id CDATA #IMPLIED>
```

```
<!ELEMENT TITULO (#PCDATA)>
<!ELEMENT EDITORA (#PCDATA)>
<!ELEMENT ANO (#PCDATA)>
]>
```

Esta DTD define que o elemento BIBLIOTECA deve conter no mínimo um LIVRO, caracterizado pelo sinal "+". Um LIVRO deve conter no mínimo um AUTOR, necessariamente um TITULO e pode ou não ter EDITORA e ANO (indicado pelo sinal "?"). Elementos terminais são declarados como

```
#PCDATA
```

(parsed character data). A tag

```
"<!ATTLIST"
```

serve para declarar o atributo "id", que é do tipo CDATA (character data) e é opcional

```
(#IMPLIED).
```

Uma DTD é declarada no prólogo do documento XML usando a tag !DOCTYPE. A DTD pode ser incluída no próprio documento XML ou em um arquivo separado. Caso esteja em arquivo separado o documento XML deve conter:

```
<!DOCTYPE Document SYSTEM "document.dtd">
```

2.3 XML Schema

Apesar de terem sido desenvolvidas como um padrão para troca de dados entre usuários, as DTDs apresentam algumas desvantagens. Seu poder de expressão é limitado e sua sintaxe não é XML. Devido a esses fatores, XML Schema vem sendo largamente utilizado em substituição a DTD.

XML Schema é um documento XML bem formado que permite que o usuário defina tipos de dados.

Exemplo do XML Schema correspondente a DTD acima:

```
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<xsd:element name="BIBLIOTECA" type="BIBLIOTECA_TYPE"/>
<xsd:complexType name="BIBLIOTECA_TYPE " >
<xsd:element name="LIVRO" type="xsd:string" minOccurs="1" maxOccurs="unbounded"/>
</xsd:complexType>
```

```
<xsd:element name="LIVRO" type="LIVRO_TYPE"/>
<xsd:complexType name="LIVRO_TYPE " >
<xsd:element name="AUTOR" type="xsd:string" minOccurs="1" maxOccurs="unbounded"/>
<xsd:element name="TITULO" type="xsd:string"/>
<xsd:element name="EDITORA" type="xsd:string" minOccurs="0" maxOccurs="1"/>
<xsd:element name="ANO" type="xsd:decimal" minOccurs="0" maxOccurs="1"/>
</xsd:complexType>
<xsd:element name="AUTOR" type="AUTOR_TYPE"/>
<xsd:complexType name="AUTOR_TYPE " >
<xsd:attribute name="id" type="xsd:string"/>
</xsd:complexType>
</xsd:schema>
```

XML Schema utiliza seu próprio namespace e suporta uma variedade de tipos de dados atômicos como string, decimal e date. O usuário pode simular os sinais +, * e ? das DTDs usando os atributos minOccurs e maxOccurs.

Por ser mais expressivo XML Schema permite definir o elemento ANO como sendo um decimal

`(<xsd:decimal>).`

2.4 XSL

XSL é uma linguagem para expressar folhas de estilo (style sheets). Uma folha de estilo XSL é, como um CSS, um arquivo que descreve como mostrar um documento XML de um certo tipo. XSL divide a funcionalidade e é compatível com CSS2 (embora use uma sintaxe diferente). Contém também:

- Uma linguagem de transformação de documentos XML: XSLT. Originalmente pretendia fornecer operações de estilo complexas, como geração de tabelas de conteúdos e índices, e é agora usada como uma linguagem de processamento XML de propósito geral. XSLT é usada largamente para outras finalidades diferentes de XSL, como geração de páginas HTML a partir de dados XML.

- Funções avançadas de estilo, expressas por um tipo de documento XML o qual define um conjunto de elementos chamados de Formatting Objects, e atributos (alguns emprestados das propriedades do CSS2 e outros mais complexos que foram adicionados).

Aplicar um estilo requer um documentos XML fonte, contendo a informação que a folha de estilo irá mostrar e a própria folha de estilo em si, a qual descreve como mostrar o documento de um determinado tipo.

Exemplo de um documento XSL:

```
<xsl:stylesheet version = '1.0'
               xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match="/">
     <h1>
          <xsl:value-of select="//nome"/>
     </h1>
     <h2>
          <xsl:value-of select="//autor"/>
     </h2>
</xsl:template>
</xsl:stylesheet>
}
```

2.5 XQuery

O Query Working Group identificou requisitos tanto para sintaxes de consulta legíveis por humanos quanto para sintaxes de consulta baseadas em XML. XQuery foi desenhada para atender o primeiro deles.

XQuery é derivada de uma linguagem de consulta XML chamada Quilt, que por sua vez se originou a partir de outras linguagens como XPath 1.0, XQL, XML-QL, SQL e OQL. XQuery versão 1.0 é uma extensão de XPath versão 2.0.

O bloco básico de uma XQuery é uma expressão. A linguagem provê vários tipos de expressões, as quais podem ser construídas por palavras-chave, símboos e operandos. No geral, os operandos de uma expressão são outras expressões. XQuery é uma linguagem funcional que permite aninhar quaisquer expressões. É fortemente tipada, portanto os operandos de uma expressão, os operadores e as funções deve estar em conformidade com os tipos designados. Como XML, XQuery é case-sensitive, ou seja, diferencia maiúsculas de minúsculas. Todas as palavras-chave em XQuery são minúsculas.

O valor de uma expressão é sempre uma seqüência, a qual é uma coleção ordenada de zero ou mais itens. Um item é um valor atômico ou um nodo. Um valor atômico é um valor no espaço de valores de um tipo atômico do XML Schema, que é um tipo simples, ou seja, não é tipo lista nem tipo union. Cada nodo possui uma identificação única.

O exemplo a seguir ilustra como pode-se expressar uma consulta que combina dados de múltiplas fontes em um único resultado utilizando XQuery. Para tanto deve-se considerar:

- Um documento chamado parts.xml que contém muitos elementos part ; cada elemento part por sua vez contém os sub-elementos partno e description .
- Um documento chamado suppliers.xml que contém muitos elementos supplier ; cada elemento supplier por sua vez contém os sub-elementos suppno e suppname .
- Um documento chamado catalog.xml que contém informação sobre os suppliers e parts. O documento catalog contém muitos elementos item , que por sua vez contêm os sub-elementos partno, suppno, e price .

Exemplo de uma consulta utilizando XQuery:

```
<descriptive-catalog>
   {
     for $i in document("catalog.xml")//item,
         $p in document("parts.xml")//part[partno = $i/partno],
         $s in document("suppliers.xml")//supplier[suppno = $i/suppno]
     order by $p/description, $s/suppname
     return
        <item>
           {
           $p/description,
           $s/suppname,
           $i/price
           }
        </item>
   }
</descriptive-catalog>
```

O exemplo combina informações de três documentos e gera um "descriptive catalog"derivado do documento catalog, mas contendo part descriptions a invés de part numbers e supplier ao invés de supplier numbers. O novo catalog é ordenado alfabeticamente por part description e secundariamente por supplier name. A consulta anterior retorna informações somente sobre parts que têm suppliers e suppliers que têm parts.

Capítulo 3

Estado da Arte

3.1 Banco de Dados

Atualmente existem no mercado vários bancos de dados proprietários que permitem consulta e armazenamento de documentos XML. Estes bancos, apesar de oferecerem funcionalidades complexas, são pagos e seu código é fechado. O usuário fica dependente do fabricante.

3.2 Projeto XEAR

Em pesquisas recentes do Projeto XEAR foi desenvolvido uma maneira de traduzir XQuery em SQL e vice-versa. Para tanto fez-se o uso de ferramentas que convertem XQuery em árvores XAT e ferramentas que a partir de uma árvore XAT extrai comandos SQL. Apesar de demonstrar o processo os algoritmos não estão disponíveis ao público.

3.3 Necessidades do Mercado

Observou-se a importância de se prover uma consulta a um banco de dados onde o retorno da mesma não fosse uma tabela e sim um documento no formato especificado pelo usuário. A idéia é que esse mapeamento seja transparente ao usuário, podendo ser feito através de um arquivo de configuração ou de variáveis em templates.

3.4 Solução Encontrada

Devido a complexidade em obter-se um comando SQL a partir de uma XQuery pensou-se em projetar um framework que fosse simples e que então permitisse qualquer tipo de consulta, ficando a cargo do desenvolvedor implementar a conversão necessária, de acordo com o tipo de consulta desejada. O framework ficaria disponível aos usuários.

Várias aplicações utilizam seu próprio banco de dados. O framework ajudaria essas aplicações a disponibilizarem seus dados em um formato XML ou outro qualquer de acordo com as necessidades.

Capítulo 4

O Framework Proposto

Como vimos anteriormente, existe uma deficiência nas aplicações e sistemas para utilização de XML com banco de dados propostos atualmente, no sentido da abrangência das aplicações que utilizarão estes recursos, e que a solução consiste em fazer com que os bancos de dados possam fornecer dados em qualquer formato XML. Porém, como fazer o banco de dados formatar os seus dados em um tipo de documento que ele não conhece? E como ele pode saber qual é a estrutura XML que o requerente dos dados espera? E mais, que alterações seriam necessárias para que sistemas de banco de dados já instalados possam se beneficiar desta nova vantagem?

A resposta para estas perguntas é: Não mexendo no banco de dados! Por que se desfazer do seu sistema preferido, transportar todos os dados e aprender a mexer em um novo sistema só para adicionar uma nova funcionalidade? Toda a parte de conversão entre estruturas pode ficar a cargo de um software separado, um middleware, que faz a ligação entre o cliente e o banco de dados. O middleware recebe uma requisição em qualquer formato de texto, analisa-a, faz a consulta necessária ao banco de dados, monta o documento de retorno no mesmo formato que recebeu e envia-o para o requerente.

Porém fica ainda a dúvida de como fazer o middleware reconhecer qualquer tipo de documento para que possa formatar a sua resposta. A solução baseia-se no mesmo conceito do script para home-pages PHP. O servidor trabalha em cima de um documento texto qualquer onde partes do texto são códigos de script PHP. Esses códigos são processados e o resultado é o próprio documento reformulado, sem os scripts e com algumas modificações feitas por eles.

A requisição recebida pelo middleware, portanto, não seria um documento XML puro, mas um que seguisse a estrutura de XML na qual o cliente trabalha mais um script que diz o que deve ser consultado no banco de dados e como o resultado deve ser formatado de modo que continue com a mesma

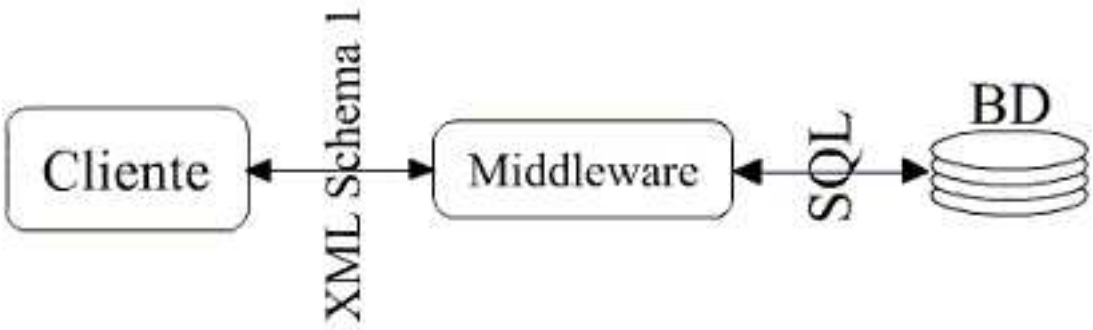

Figura 4.1: Transação Cliente-Middleware-BD

estrutura. O documento de retorno seria o próprio documento recebido, sem os scripts e com as alterações definidas no código.

Nota-se até aqui a simplicidade do processo. Não é necessário trabalhar com documentos que especifiquem toda a estrutura dos documentos XML envolvidos (DTD, XML Schema, ...), muito menos ter que definir como deve ser o mapeamento dos dados contidos em cada tabela, de um banco de dados relacional por exemplo, para os elementos, entidades, atributos e valores do documento XML requisitado (esse é um dos maiores problemas enfrentados hoje em dia quando a complexidade do banco de dados é muito grande).

4.1 Requisitos

Durante o projeto do framework foram identificados alguns requisitos básicos. São eles:

- O framework funcionará como um middleware que fica entre um banco de dados qualquer e as aplicações que precisam consultar este banco.
- As aplicações cliente enviarão a consulta para o middleware que se encarregará de coletar os dados necessários no banco de dados e formatar a devida resposta.
- As consultas serão de modo textual, seguindo o padrão da maioria dos protocolos em evidência (HTTP, SMTP, SOAP,...).
- Não deve existir nenhuma restrição ou dependência sintática nos textos de consulta e de resposta.
- Tornar os módulos tão maleáveis, genéricos e enxutos quanto possível.

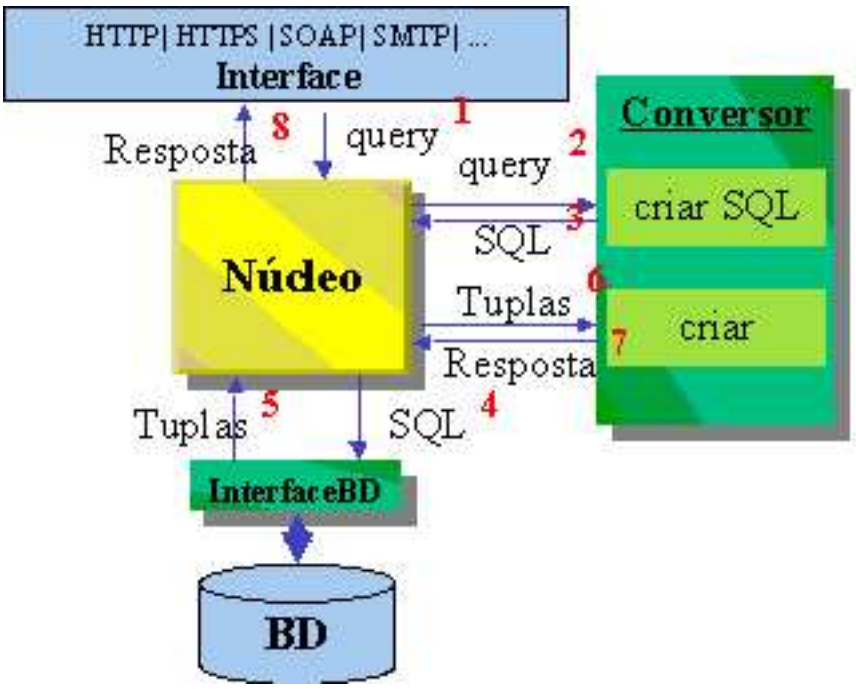

Figura 4.2: Arquitetura do Framework

4.2 Arquitetura

O framework apresenta 4 módulos distintos:

- Interface: Este módulo deve implementar toda a parte de comunicação com as aplicações cliente. É aqui que o desenvolvedor definirá coisas do tipo: como será feita a comunicação (linha de comando, pipes, sockets, ...), que protocolo usar (HTTP, SMTP,...) e segurança (permissões de acesso, encriptação). Este é um módulo totalmente abstrato e não tem uma uma classe específica no modelo, uma vez que a sua implementaçao depende muito do tipo de comunicação externa.

- Conversão: Encarregado de criar tanto o comando SQL para ser enviado ao DBMS quanto o documento de resposta que retornará ao cliente. A interface envia a consulta recebida para o conversor, este então criará os comandos SQL para resgatar os dados necessários do banco de dados. Assim que os dados forem coletados, eles também serão encaminhados para este módulo que criará a resposta para o cliente no formato esperado. É no módulo de conversão que o desenvolvedor definirá as sintaxes das consultas e dos documentos de retorno.

- Interface com DBMS: Cuida da comunicação com o DBMS. Este módulo envia os comandos SQL gerados pelo módulo conversor para o DBMS retornando a tuplas, caso existam.
- Kernel: É o módulo que controla o fluxo de execução da consulta e é aquele com o qual o resto da aplicação que usará este framework trabalhará. Seu funcionamento básico consiste num laço com as seguintes operações:
  - Converter a consulta recebida em uma linguagem conhecida pelo DBMS (módulo Converter)
  - Envio da consulta para o DBMS (módulo DbInterface)
  - Conversão das tuplas para o ducumento esperado (módulo Converter)

  Estas operações são repetidas até que todas as consultas ao banco de dados, necessárias para a formação do documento de resposta, sejam processadas. Este módulo é representado por uma clases cujo processo descrito já se encontra implementado e que servirá para a maioria das aplicações.

4.3 Diagramas

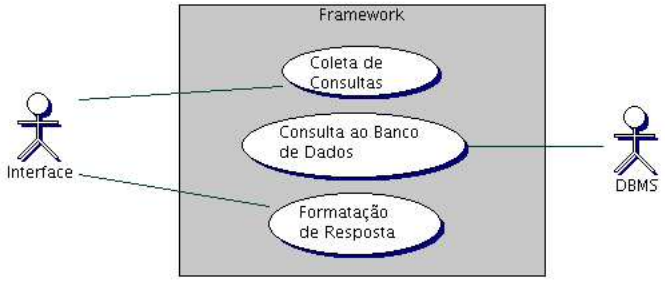

Figura 4.3: Diagrama de Use Case do Framework

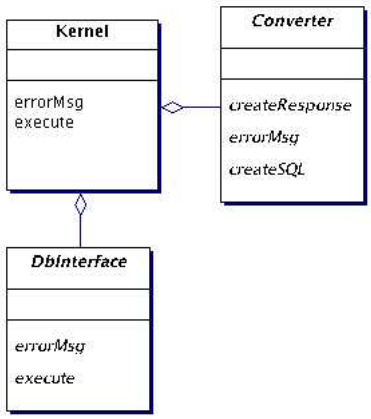

Figura 4.4: Diagrama de Classes do Framework

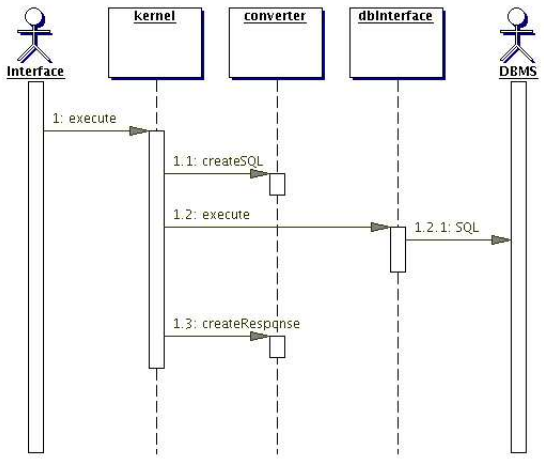

Figura 4.5: Diagrama de Sequência do Framework

Capítulo 5

O Protótipo - Validação do Framework

Para fazer a validação do framework para consultas genéricas a banco de dados relacional foi escolhida uma aplicação cliente, baseada na idéia original de se obter um documento XML como resposta à consulta de outro documento XML a um banco de dados relacional.

Uma das etapas mais importantes deste projeto consiste na definição do script que estipula quais são os dados que o cliente deseja e como devem ser formatados para que o documento final pertença à estrutura esperada. Portanto podemos definir duas etapas para o script: consulta e formatação.

Para a consulta, precisamos definir como o cliente pedirá os dados que precisa. A solução mais simples que se pode imaginar é utilizar SQL. Esta é uma linguagem que se tornou universal na área de banco de dados relacionais e, devido a sua grande aceitação, é utilizada até mesmo com outro tipos de bancos, como os hierárquicos e orientados a objetos. Utilizando-se SQL, a tarefa do middleware de retransmitir a consulta do cliente para o banco de dados se torna trivial. Basta repassar o próprio código SQL. Não é necessário nenhum tradutor especial.

Seguindo sempre idéia de simplicidade, para a parte de formatação são usadas variáveis, onde cada uma representa uma coluna da tabela resultante da etapa de consulta, e templates, que terão suas variáveis preenchidas uma vez para cada linha da mesma tabela.

Para entender melhor o conceito vamos analisar um exemplo prático. Digamos que um sistema contábil de supermercado deseja consultar em seu sistema o preço dos sabonetes disponíveis. Ele poderia então enviar para o middleware o seguinte documento XML:

```
<?xml version="1.0" standalone="yes" ?>
```

```
<!DOCTYPE CONTABILIDADE SYSTEM "contabilidade.dtd">
<PRODUTOS>
  <?SQL
     SELECT marca, preço FROM produtos WHERE tipo='sabonete'
  SQL?>
  <?TEMPL
    <PRODUTO tipo = \SABONETE">
      <MARCA>$_marca</MARCA>
      <PREÇO>R$$_preço</PREÇO>
    </PRODUTO>
  TEMPL?>
</PRODUTOS>
```

O middleware por sua vez, pegaria o código SQL contido entre as tags

```
<?SQL e SQL?>
```

e o enviaria ao banco de dados que responderia com a seguinte tabela:

Marca	Preço
Marca X	1,20
Marca Y	1,00
Marca Z	1,35

Com esta tabela em mãos, o middleware poderá formatar a resposta repetindo o template definido entre as tags

```
 <?TEMPL e TEMPL?>
```

tantas vezes quanto o número de linhas existentes na tabela e substituindo as variáveis, definidas por um

```
"$_"
```

inicial, pelos valores correspondentes em cada coluna. O documento XML resultante seria então:

```
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE CONTABILIDADE SYSTEM "contabilidade.dtd">
<PRODUTOS>
  <PRODUTO tipo = \SABONETE">
    <MARCA>Marca X</MARCA>
    <PREÇO>R$1,20</PREÇO>
  </PRODUTO>
```

```
  <PRODUTO tipo = \SABONETE">
    <MARCA>Marca Y</MARCA>
    <PREÇO>R$1,00</PREÇO>
  </PRODUTO>
  <PRODUTO tipo = \SABONETE">
    <MARCA>Marca Z</MARCA>
    <PREÇO>R$1,35</PREÇO>
  </PRODUTO>
</PRODUTOS>
```

Assim, o documento resultante segue exatamente a estrutura esperada pelo sistema contábil (contabilidade.dtd) e poderá ser enviado de volta para ele.

Capítulo 6

Conclusão e Resultados

Com a implementação do protótipo, que fez uso do framework aqui apresentado, conclui-se que o emprego das técnicas de análise de sistemas e engenharia de software são fatores importantes no desenvolvimento de novas tecnologias.

O framework é o encarregado pelos aspectos básicos e comuns relativos às aplicações que acessam bancos de dados e exigem algum tipo de processamento da consulta e/ou resposta, e faz com que o desenvolvedor foque sua concentração nos detalhes específicos somente ao seu problema.

Capítulo 7

Trabalhos Futuros

Há muito o que se implementar quando o assunto é facilitar consultas a bancos de dados principalmente quando o foco é a transparência da operação. Tendo como base o framework implementado e seguindo a tendência do mercado e as reais necessidades do mesmo, sugere-se algumas linhas de desenvolvimento. São elas:

- Implementar consultas XQuery e a devida conversão para SQL, utilizando o framework proposto
- Implementar apresentação do documento final baseando-se na estrutura de uma DTD ou de um XML Schema
- Fazer o mapeamento, podendo utilizar um arquivo de configuração para tanto, entre os elementos de um documento XML e a tabela do banco de dados correspondente ao mesmo.
- Utilizar banco de dados orientado a objetos.
- Implementação do framework e do protótipo utilizando software livre.

Bibliografia

[1] André (Stanford University) Bergholz. Extending your markup: An xml tutorial, 2000. [http://computer.org/internet/xml/xml.tutorial.pdf] (Out 2, 2002).

[2] Ronald Bourret. Xml programming, writing and research - informações sobre xml voltado à banco de dados. *http://www.rpbourret.com*, 2002.

[3] Roger L. Costello. Xml schema tutorial. [http://www.xfront.com/xml-schema.html] (Nov 17, 2002).

[4] Laboratório de Ciências da Computação do MIT. World Wide Web Consortium. Online, 1994-2003. [http://www.w3c.org] (Abr 27, 2002).

[5] XML Query Working Group. Xquery tutorial. [http://www.w3.org/TR/xquery/] (Jan 3, 2003).

[6] Proschitsky A. Lutz J. E., Pielech B. C. Xear: Xquery processor with relational databases. *http://davis.wpi.edu/dsrg/WEB_DB/XQuery/*, 2002.

[7] Michael Young. *XML - Step by Step*. Microsoft Editora, 2000.

Apêndice A

Código Fonte do Framework

```
/* UDm.pas */

unit UDm;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  ScktComp, UTKernel, Db, DBTables, syncobjs;

type
  Tdm = class(TDataModule)
    ServerSocket: TServerSocket;
    query: TQuery;
    procedure DataModuleCreate(Sender: TObject);
    procedure DataModuleDestroy(Sender: TObject);
    procedure ServerSocketGetThread(Sender: TObject;
      ClientSocket: TServerClientWinSocket;
      var SocketThread: TServerClientThread);
  private
    { Private declarations }
  public
    { Public declarations }
    semaphore: TCriticalSection;
  end;

  { TFileServerThread }
```

```
  TQueryServerThread = class(TServerClientThread)
  public
    procedure ClientExecute; override;
  end;

var
  dm: Tdm;

implementation

uses
  UTMyConverter;

{$R *.DFM}

procedure Tdm.DataModuleCreate(Sender: TObject);
begin
  semaphore:= TCriticalSection.Create;
  query.OpenDatabase;
end;

procedure Tdm.DataModuleDestroy(Sender: TObject);
begin
  semaphore.Free;
end;

procedure Tdm.ServerSocketGetThread(Sender: TObject;
  ClientSocket: TServerClientWinSocket;
  var SocketThread: TServerClientThread);
begin
  socketThread:= TQueryServerThread.Create(false,ClientSocket);
end;

{ TQueryServerThread }

procedure TQueryServerThread.ClientExecute;
var
  buffer: array[0..100000] of char;
  query, xmlResult: string;
  totalLen,len, error: integer;
```

```
  SocketStream: TWinSocketStream;
begin
  SocketStream := TWinSocketStream.Create(ClientSocket, 3000);
  try
    write('Conexao com IP: '+ClientSocket.RemoteAddress+#13#10);
    fillchar(buffer,sizeof(buffer),0);
    totalLen:= 0;
    repeat
      len:= socketStream.Read(buffer[totalLen],sizeof(buffer)-totalLen);
      inc(totalLen,len);
    until (len = 0) or (totalLen >= sizeof(buffer));

    if totalLen > 0 then begin
      query:= buffer;
      error:= kernel.execute(query,xmlResult);
      if error = 0 then begin
        xmlResult:= '0'#13#10+xmlResult;
        fillchar(buffer,sizeof(buffer),0);
        for len:= 0 to length(xmlresult)-1 do
buffer[len]:= xmlResult[len+1];
        socketStream.Write(buffer,length(xmlResult));
      end
      else begin
        xmlResult:= inttostr(error)+#13#10+kernel.errorMsg(error);
        fillchar(buffer,sizeof(buffer),0);
        for len:= 0 to length(xmlresult)-1 do
buffer[len]:= xmlResult[len+1];
        socketStream.Write(buffer,length(xmlResult));
      end;
    end;

  finally
    socketStream.free;
    Terminate;
  end;
end;

end.


/* UTConverter.pas */
```

```
unit UTConverter;

interface

type
  TConverter = class
    function createSQL (var query:string;
              var sqlResult: string): integer; virtual; abstract;
    function createResponse (tuples: pointer; var Response: string;
               var continue:boolean): integer; virtual; abstract;
    function errorMsg (errorNum: integer): string; virtual; abstract;
    //error numbers must be between [100,199]
  end;

implementation

end.


/* UTDbInterface.pas */

unit UTDbInterface;

interface

type
  TDbInterface = class
    function execute (sql: string; var tuples: pointer): integer;
virtual; abstract;
    function errorMsg (errorNum: integer): string; virtual; abstract;
    //error numbers must be between [200,299]
  end;

implementation

end.


/* UTKernel.pas */
```

```
unit UTKernel;

interface
uses UTDbInterface, UTConverter;

type
  converterClass = class of TConverter;
  DbInterfaceClass = class of TDbInterface;

  TKernel = class
  protected
    converter: TConverter;
    dbInterface: TDbInterface;
  public
    constructor create (converterC: converterClass;
                        dbInterfaceC: DbInterfaceClass);
    destructor destroy; override;
    function execute (query:string; var response: string): integer;
virtual;
    function errorMsg (errorNum: integer): string ; virtual;
  end;

var
  kernel: TKernel;

implementation

uses  UDm;

{ TKernel }

constructor TKernel.create(converterC: converterClass;
                        dbInterfaceC: DbInterfaceClass);
begin
  converter:= converterC.Create;
  dbInterface:= dbInterfaceC.Create;
end;

destructor TKernel.destroy;
begin
  converter.free;
```

```
  dbInterface.free;
  inherited;
end;

function TKernel.errorMsg(errorNum: integer): string;
begin
  case errorNum of
    0: result:= 'Ok';
    100..199 : result:= converter.errorMsg(errorNum);
    200..299 : result:= dbInterface.errorMsg(errorNum);
    else result:= 'Error not defined';
  end;
end;

function TKernel.execute(query:string; var response: string): integer;
var
  sql: string;
  tuples: pointer;
  continue: boolean;
begin
  repeat
    result:= converter.createSQL(query,sql);
    if result <> 0 then exit;
    result:= dbInterface.execute(sql,tuples);
    if result <> 0 then exit;
    result:= converter.createResponse(tuples,response,continue);
  until not continue;
end;

end.
```

Apêndice B

Código Fonte do Protótipo

```
/* queries.txt */
/* arquivo de requisição ao servidor */

<locadora>
  <?sql select CID,NOME,TELEFONE from clientes sql?>
  <clientes>
  <?templ
    <cliente>
      <cid>$cid</cid>
      <nome>$nome</nome>
      <telefone>$telefone</telefone>
      <?sql select TITULO
            from fitas f, locacoes l
            where l.tid = f.tid AND l.cid = $cid sql?>
      <locacoes>
      <?templ
        <titulo>$titulo</titulo>
      templ?>
      </locacoes>
    </cliente>
  templ?>
  </clientes>


  <fitas>
  <?sql select tid, titulo, data_aquisicao, capa from fitas sql?>
  <?templ
    <fita>
```

```
      <tid>$tid</tid>
      <titulo>$titulo</titulo>
      <aquisicao>$data_aquisicao</aquisicao>
      <capa href"$capa" />
    </fita>
  templ?>
  </fitas>
</locadora>


/* UTMyKernel.pas */

unit UTMyKernel;

interface
uses UTDbInterface, UTConverter, UTKernel;

type
  TMyKernel = class(TKernel)
  public
    function execute (query:string; var response: string): integer;
override;
  end;

implementation


{ TMyKernel }

function TMyKernel.execute(query: string; var response: string):
integer;
var
  sql: string;
  tuples: pointer;
  continue: boolean;
begin
  repeat
    result:= converter.createSQL(query,sql);
    response:= query;
    if result <> 0 then exit;
    result:= dbInterface.execute(sql,tuples);
```

```
    if result <> 0 then exit;
    result:= converter.createResponse(tuples,response,continue);
    query:= response;
  until not continue;
end;

end.


/* UTMyDbInterface.pas */

unit UTMyDbInterface;

interface

uses UTDbInterface, dbtables, Sysutils;

type
  TMyDbInterface = class(TDbInterface)
  private
    errorStr: string;
  public
    function execute (sql: string; var tuples: pointer): integer;
override;
    function errorMsg (errorNum: integer): string; override;
    //error numbers must be between [200,299]
  end;

implementation

uses UDm;

{ TMyDbInterface }


function TMyDbInterface.errorMsg(errorNum: integer): string;
begin
  case errorNum of
    200: result:= errorStr;
    else result:= 'Error not defined';
  end;
```

```
end;

function TMyDbInterface.execute(sql: string; var tuples: pointer):
integer;
begin
  try
    dm.query.close;
    dm.query.SQL.Text:= sql;
    dm.query.open;

    tuples:= dm.query;
    result:= 0;
  except
    on E: Exception do begin
      result:= 200;
      errorStr:= E.Message;
    end;
  end;
end;

end.


/* UTMyConverter.pas */

unit UTMyConverter;

interface
uses UTConverter, SysUtils, dbtables, UDm;

type
  TMyConverter = class (TConverter)
  public
    function createSQL (var query:string;
                        var sqlResult: string): integer; override;
    function createResponse (tuples: pointer;
                             var Response: string;
                             var continue:boolean): integer; override;
    function errorMsg (errorNum: integer): string; override;
    //error numbers must be between [100,199]
  end;
```

```
implementation

uses Db, graphics, classes, jpeg;

const
  fileID: byte = 0;
  filesPath = 'C:/Apache/htdocs/prototipo/';
  filesAddress = 'prototipo/';

{ TConverter }

function TMyConverter.createResponse (tuples: pointer;
                              var Response: string;
                              var continue:boolean): integer;
var
  querier: TQuery;
  i, f, k, count: integer;
  respBegin,respEnd,template,filled,
  fieldName,
  fileName: string;
  field: TField;
  bmp: TBitmap;
  jpg: TJPEGImage;
begin
  result:= 0;

  i:= pos('<?templ',Response);
  if i = 0 then begin
    continue:= false;
    exit;
  end;
  f:= i+8;
  count:= 0;
  k:= length(response);
  while f < k do begin
    if response[f] = '<' then begin
      if copy(response,f,7) = '<?templ' then begin
        inc(count);
        inc(f,7);
      end;
```

```
    end
    else if response[f] = 't' then begin
      if copy(response,f,7) = 'templ?>' then begin
        if count = 0 then break
        else dec(count);
      end;
    end;
    inc(f);
  end;
  if f > k then begin
    result:= 103;
    exit;
  end;

  respBegin:= copy(Response,1,i-1);
  respEnd  := copy(Response,f+7,99999999999);
  inc(i,7);
  template:= copy(Response,i,f-i);

  Response:= respBegin;
  querier:= TQuery(Tuples);
  querier.First;
  while not querier.Eof do begin
    filled:= template;
    for f:=0 to querier.Fields.Count-1 do begin
      field:= querier.Fields[f];
      fieldName:= field.FieldName;
      if field.IsBlob then begin
        bmp:= TBitmap.create;
        jpg:= TJpegImage.Create;
        try
          fileName:= inttostr(fileId);
          inc(fileID);
          fileName:= fileName + '.jpg';
          filled:= stringReplace(filled,'$'+fieldName,filesAddress
+ fileName,[rfReplaceAll,rfIgnoreCase]);
          bmp.Assign(TBlobField(field));
          jpg.Assign(bmp);
          jpg.SaveToFile(filesPath + fileName);
        finally
          jpg.free;
```

```
          bmp.free;
        end;
      end
      else
        filled:= stringReplace(filled,'$'+fieldName,field.AsString,
[rfReplaceAll,rfIgnoreCase]);
    end;
    Response:= Response + filled;
    querier.Next;
  end;
  Response:= Response + respEnd;

  continue:= pos('<?sql',response)>0;
end;

function TMyConverter.createSQL (var query:string;
                        var sqlResult: string): integer;
var
  i,f: integer;
  sql: string;
begin
  result:= 0;
  i:= pos('<?sql',query);
  if i = 0 then begin result:= 100; exit; end;
  f:= pos('sql?>',query);
  if f < i then begin result:= 101; exit; end;

  inc(i,5);
  sql:= trim(copy(query,i,f-i));
  if sql = '' then begin result:= 102; exit; end;

  dec(i,5);
  delete(query,i,f-i+5);

  sqlResult:= sql;
end;

function TMyConverter.errorMsg(errorNum: integer): string;
begin
  case errorNum of
    100: result:= 'No SQL tag found';
```

```
    101: result:= 'Incomplete SQL tag';
    102: result:= 'No SQL statement found';
    103: result:= 'Incomplete template tag';
    else result:= 'Error not defined';
  end;
end;

end.


/* queryd.dpr */

program queryd;

uses
  Forms,
  UDm in 'UDm.pas' {dm: TDataModule},
  UTKernel in 'UTKernel.pas',
  UTDbInterface in 'UTDbInterface.pas',
  UTMyDbInterface in 'UTMyDbInterface.pas',
  UTConverter in 'UTConverter.pas',
  UTMyConverter in 'UTMyConverter.pas',
  UTMyKernel in 'UTMyKernel.pas';

{$R *.RES}


begin
  Application.Initialize;
  Application.CreateForm(Tdm, dm);
  dm.ServerSocket.Open;
  kernel:= TMyKernel.create(TMyConverter,TMyDbInterface);

  while true do application.ProcessMessages;
  kernel.free;
end.
```

Apêndice C

Apresentação dos Dados do Protótipo

```
/* Locadora.xsl */

<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
<html>
<head>
<title>Dados da Locadora</title>
<style type="text/css">
table {  font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 10px}
</style>
</head>

<body bgcolor="#0099FF" text="#FFFFFF">

<xsl:if test="locadora/clientes/cliente">
<p><font size="6"><u><b>Clientes</b></u></font></p>

<table width="100%" border="0" cellspacing="0" cellpadding="1"
bordercolor="#0099FF" bgcolor="#0066FF" align="center">
  <tr>
    <td width="7%" bgcolor="#0099FF"><img src="id.gif"/></td>
    <td width="46%" bgcolor="#0099FF"><img src="nome.gif"/></td>
```

```
    <td width="18%" bgcolor="#0099FF"><img src="telefone.gif"/></td>
    <td width="29%" bgcolor="#0099FF"><img src="locacoes.gif"/></td>
  </tr>
<xsl:for-each select="locadora/clientes/cliente">
  <xsl:if test="position() mod 2 =1">
    <tr align="left" valign="top" bgcolor="#0088FF">
      <td width="7%" ><xsl:value-of select="cid"/></td>
      <td width="46%"><xsl:value-of select="nome"/></td>
      <td width="18%"><xsl:value-of select="telefone"/></td>
      <td width="29%">
        <xsl:for-each select="locacoes/titulo">
          <xsl:value-of select="text()"/><br/>
        </xsl:for-each>
      </td>
    </tr>
  </xsl:if>
  <xsl:if test="position() mod 2 = 0">
    <tr align="left" valign="top">
      <td width="7%" ><xsl:value-of select="cid"/></td>
      <td width="46%"><xsl:value-of select="nome"/></td>
      <td width="18%"><xsl:value-of select="telefone"/></td>
      <td width="29%">
        <xsl:for-each select="locacoes/titulo">
          <xsl:value-of select="text()"/><br/>
        </xsl:for-each>
      </td>
    </tr>
  </xsl:if>
</xsl:for-each>
</table>

<p><br/></p>
</xsl:if>

<xsl:if test="locadora/fitas/fita">
<p><font size="6"><u><b>Fitas</b></u></font></p>

<table width="100%" border="0" cellspacing="0" cellpadding="1"
bordercolor="#0099FF"
bgcolor="#0066FF" align="center">
  <tr>
```

```
    <td width="7%" bgcolor="#0099FF"><img src="id.gif"/></td>
    <td width="41%" bgcolor="#0099FF"><img src="titulo.gif"/></td>
    <td width="23%" bgcolor="#0099FF"><img src="aquisicao.gif"/></td>
    <td width="29%" bgcolor="#0099FF"><img src="capa.gif"/></td>
  </tr>
<xsl:for-each select="locadora/fitas/fita">
  <xsl:if test="position() mod 2 =1">
    <tr align="left" valign="top" bgcolor="#0088FF">
      <td width="7%" ><xsl:value-of select="tid"/></td>
      <td width="41%"><xsl:value-of select="titulo"/></td>
      <td width="23%"><xsl:value-of select="aquisicao"/></td>
      <td width="29%">
        <img>
          <xsl:attribute name="src">
            <xsl:value-of select="capa/attribute::href"/>
          </xsl:attribute>
        </img>
      </td>
    </tr>
  </xsl:if>
  <xsl:if test="position() mod 2 = 0">
    <tr align="left" valign="top">
      <td width="7%" ><xsl:value-of select="tid"/></td>
      <td width="41%"><xsl:value-of select="titulo"/></td>
      <td width="23%"><xsl:value-of select="aquisicao"/></td>
      <td width="29%">
        <img>
          <xsl:attribute name="src">
            <xsl:value-of select="capa/attribute::href"/>
          </xsl:attribute>
        </img>
      </td>
    </tr>
  </xsl:if>
</xsl:for-each>
</table>
</xsl:if>
</body>
</html>

</xsl:template>
```

```
</xsl:stylesheet>


/* Página Web */

/* index.html */

<html>
<head>
<title>Locadora Prototipe / P&aacute;gina de Consulta</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
<style type="text/css">
<!--
body {  font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 10pt}
table {  font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 10pt}
-->
</style>
</head>

<body bgcolor="#0099FF" text="#FFFFFF">
<p><img src="logo.gif" width="263" height="96"> </p>
<table width="512" border="0" cellspacing="0" cellpadding="3"
bgcolor="#00CCFF">
  <tr>
    <td height="20"> <font color="#0000FF" size="5">Consulta</font>
      <form name="form1" method="post" action="consulta.php">
        <table width="100%" border="0" cellspacing="2"
cellpadding="3" bgcolor="#00CCFF" height="324">
          <tr>
            <td bgcolor="#0066FF" height="70"> <b>Clientes</b><br>
              <br>
              <table width="484" border="0" cellspacing="0"
cellpadding="3" align="center">
                <tr>
                  <td width="62" height="2">
                    <div align="right">ID</div>
                  </td>
                  <td width="260" height="2">
```

```
                    <input type="text" name="tfCid" size="5"
maxlength="5" value="*">
                    <font size="1">*, =, &gt;, &lt;, &gt;=, &lt;=,
&lt;&gt;</font></td>
                  <td width="144" height="2">  </td>
                </tr>
                <tr>
                  <td width="62" height="2">
                    <div align="right">Nome</div>
                  </td>
                  <td width="260" height="2">
                    <input type="text" name="tfNome" size="50"
maxlength="100"> </td>
                  <td width="144" height="2"> <font size="1">
nome completo </font></td>
                </tr>
                <tr>
                  <td width="62" height="2">
                    <div align="right">Loca&ccedil;&ocirc;es</div>
                  </td>
                  <td width="260" height="2">
                    <input type="text" name="tfLocacao" size="50"
maxlength="250"></td>
                  <td width="144" height="2"><font size="1">
separadas por v&iacute;rgula</font></td>
                </tr>
              </table>
              <br>
            </td>
          </tr>
          <tr>
            <td bgcolor="#0066FF" height="171"><b>Fitas</b><br>
              <br>
              <table width="459" border="0" cellspacing="0"
cellpadding="3" align="center">
                <tr>
                  <td width="62" height="2">
                    <div align="right">ID</div>
                  </td>
                  <td width="260" height="2">
                    <input type="text" name="tfTid" size="5"
```

```
maxlength="5" value="*">
                    <font size="1"> *, =, &gt;, &lt;, &gt;=, &lt;=,
&lt;&gt; </font></td>
                  <td width="119" height="2">  </td>
                </tr>
                <tr>
                  <td width="62" height="2">
                    <div align="right">T&iacute;tulo</div>
                  </td>
                  <td width="260" height="2">
                    <input type="text" name="tfTitulo" size="50"
maxlength="100"></td>
                  <td width="119" height="2"><font size="1">
nome completo</font></td>
                </tr>
                <tr>
                  <td width="62">
                    <div align="right">Aquisi&ccedil;&atilde;o</div>
                  </td>
                  <td width="260">
                    <input type="text" name="tfAquisicao" size="50"
maxlength="250"></td>
                  <td width="119"><font size="1">dd/mm/aaa </font></td>
                </tr>
              </table>
              <br>
            </td>
          </tr>
        </table>
        <input type="submit" name="Submit" value="Consultar">
      </form>
    </td>
  </tr>
</table>
<p> </p>
</body>
</html>

/* Página de Consulta */
/* consulta.php */
```

```
<?php
header ("Content-Type: text/xml");

$query = "<locadora>\n";
if ((strcmp($tfCid,"")!= 0) || (strcmp($tfNome,"")!= 0) ||
(strcmp($tfLocacao,"")!= 0)) {
  if (strcmp($tfCid,"") != 0){
    if (strcmp($tfCid,"*") == 0) $query .= "<?sql SELECT cid,nome,
telefone FROM clientes ORDER BY cid sql?>";
    else {$query .= "<?sql SELECT cid,nome,telefone FROM clientes
                          WHERE cid $tfCid ORDER BY cid sql?>";}
  }
  else if (strcmp($tfNome,"") != 0){
    if (strcmp($tfNome,"*") == 0) $query .= "<?sql SELECT cid,nome,
telefone FROM clientes ORDER BY cid sql?>";
    else $query .= "<?sql SELECT cid,nome,telefone FROM clientes
WHERE nome = '$tfNome'ORDER BY cid sql?>";
  }
  else if (strcmp($tfLocacao,"") != 0){
    $query .= "<?sql SELECT cid,nome,telefone FROM clientes c, fitas f,
locacoes l WHERE l.tid = f.tid AND l.cid = c.cid AND (";
    $tfLocacao = stripslashes($tfLocacao);
    $titulos = split(",",$tfLocacao);
    while (list($key,$titulo) = each($titulos)){
      $titulo = trim($titulo);
      if (strcmp($titulo,"") != 0)
        $query .= "(f.titulo = '$titulo') OR ";
    }
    $query .= "(1=0)) ORDER BY cid sql?>";
  }

  $query .= <<<EOD

  <clientes>

  <?templ  <cliente>
      <cid>\$cid</cid>
      <nome>\$nome</nome>
      <telefone>\$telefone</telefone>
      <?sql select TITULO
            from fitas f, locacoes l
```

```
            where l.tid = f.tid AND l.cid = \$cid sql?><locacoes>
      <?templ <titulo>\$titulo</titulo>
      templ?></locacoes>
    </cliente>

  templ?></clientes>

EOD;
}

if ((strcmp($tfTid,"")!= 0) || (strcmp($tfTitulo,"")!= 0) ||
(strcmp($tfAquisicao,"")!= 0))
{
  if (strcmp($tfTid,"") != 0){
    if (strcmp($tfTid,"*") == 0) $query .= "<?sql SELECT tid, titulo,
data_aquisicao, capa FROM fitas ORDER BY tid sql?>";
    else $query .= "<?sql SELECT tid, titulo, data_aquisicao, capa
FROM fitas WHERE tid $tfTid ORDER BY tid sql?>";
  }
  else if (strcmp($tfTitulo,"") != 0){
    if (strcmp($tfTitulo,"*") == 0) $query .= "<?sql SELECT tid,
titulo,data_aquisicao, capa FROM fitas ORDER BY tid sql?>";
    else $query .= "<?sql SELECT tid, titulo, data_aquisicao, capa
FROM fitas WHERE titulo = '$tfTitulo' ORDER BY tid sql?>";
  }
  else if (strcmp($tfAquisicao,"") != 0){
    if (strcmp($tfAquisicao,"*") == 0) $query .= "<?sql SELECT tid,
titulo,data_aquisicao,capa FROM fitas ORDER BY tid sql?>";
    else $query .= "<?sql SELECT tid, titulo, data_aquisicao, capa
FROM fitas WHERE data_aquisicao = '$tfAquisicao' ORDER BY tid sql?>";
  }

$query .= <<<EOD

  <fitas>
  <?templ
    <fita>
      <tid>\$tid</tid>
      <titulo>\$titulo</titulo>
      <aquisicao>\$data_aquisicao</aquisicao>
      <capa href='\$capa' />
```

```
    </fita>
  templ?>
  </fitas>

EOD;
}
$query .= "</locadora>\n";

$sk = fsockopen("localhost",1530);
if (!$sk){
  echo "Erro ao conectar com o servidor XML.";
  exit();
}

fputs($sk,$query);
$result = fread($sk,100000);
fclose($sk);

echo "<?xml version='1.0' encoding='ISO-8859-1'?>\n";
echo "<?xml-stylesheet type='text/xsl' href='locadora.xsl'?>\n";
$result= strstr($result,"\n");

echo $result;


?>
```