UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Aplicação de XML para Estruturação de Ambientes de Controle Acadêmico baseado em Ontologias

por

CARLA DE ALMEIDA MARTINS BASSO

Dissertação submetida à avaliação
como requisito parcial para
a obtenção do grau de Mestre em
Ciência da Computação

Prof. José Palazzo Moreira de Oliveira
Orientador

Porto Alegre, novembro de 2002.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Basso, Carla de Almeida Martins

Aplicação de XML para Estruturação de Ambientes de Controle Acadêmico baseado em Ontologias / por Carla de Almeida Martins Basso. Porto Alegre: PPGC da UFRGS, 2002,

103f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR – RS, 2002. Orientador: Oliveira, José Palazzo Moreira de.

1. WWW. 2. Web. 3. Internet. 4. XML. I. Oliveira, José Palazzo Moreira de. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Prfª Wrana Panizzi
Pró-Reitor de Ensino: Prof. José Carlos Ferraz Hennemann
Pró-Reitor Adjunto de Pós-Graduação: Prof. Jaime Evaldo Fensterseifer
Diretor do Instituto de Informática: Prof. Philippe Olivier Alexandre Navaux
Coordenador do PPGC: Prof. Carlos Alberto Heuser
Bibliotecária-Chefe dói Instituto de Informática: Beatriz Regina Bastos Haro

Agradecimentos

Agradeço ao Prof. Palazzo pela compreensão, incentivo e confiança depositada durante este período.

Agradeço aos meus pais por terem permitido que eu chegasse até aqui, através do seu amor e confiança que sempre me transmitiram.

Agradeço de uma forma muito especial ao meu marido que sempre compreendeu e colaborou para o êxito deste trabalho, sempre com uma palavra de carinho.

Agradeço e dedico este trabalho aos meus dois filhos, Roberta e Pedro que mesmo em sua infância são a luz de minha vida, é por eles que aqui estou, luto e vivo cada dia com muita paixão.

Agradeço a todos os meus amigos e familiares, que através de um simples gesto ou palavra me deram esperança de vida e de futuro em todos os momentos.

Sumário

Lista de Abreviaturas 06
Lista de Figuras.. 07
Lista de Tabelas 09
Resumo 10
Abstract 11
1 Introdução 12
2 XML 14
2.1 Estrutura de um documento XML 17
2.2 As principais diferenças entre XML e HTML 19
2.3 Sintaxe básica da linguagem XML 20
2.3.1 Elementos 20
2.3.2 Atributos 20
2.3.3. Entidades 21
2.3.4. Caracteres de Dados e Marcação 22
2.3.5. Manipulação de Espaços em Branco 22
2.3.6. Comentários 22
2.3.7. Seções CDATA 23
2.3.8 Instruções de Processamento 23
2.3.9. Dados de Caracteres 23
2.3.10 Declarações. 24
2.3.11 Documentos Válidos e Bem Formados 29
2.4 Classes de Documentos XML 30
2.4.1 Document Type Definition (DTD) 31
2.4.2 XML Schema 31
2.5 Namespaces 35
2.6 Apresentação de um Documento XML 36
2.6.1 CSS 37
2.6.2 XSL 38
2.6.3 DSSSL 40
3 Processamento de Informações XML 41
3.1 Editores XML 41
3.2 Analisadores Sintáticos XML 41
3.3. Parsers 42
3.3.1 A Arquitetura de um Programa XML 42
3.4 Processadores XML 43
4 Aplicativos XML 48
4.1 Aplicativos para a Web e para a Internet 49
4.2 Aplicativos de Metadados e de Arquivamento 50
4.3 Aplicativos Multimídia 51
4.4 Aplicativos voltados às Áreas de negócios e finanças/comércio 51
4.5. Aplicativos Científicos 52
4.6 Aplicativos Educacionais 53
4.7 Aplicativos voltados a linguagens 53

5 Linguagens de Consulta para XML 55
5.1 XML-GL 56
5.2 LOREL e OQL.....58
5.3 XQL 61
5.4 WebSQL 63
5.5 WebOQL 64
5.6 XQuery 66
5.7 XML-QL.....67
6 Ontologias 70
6.1. Conceito e Características 70
6.2. Extração de Dados baseada em Ontologias 72
6.3. Aplicações 74
6.4. Aplicação de Ontologias a BDSE 74
6.4.1 O Sistema de Informação Global Observer 75
6.4.2 Linguagem SHOE 75
6.4.3 Ontobroker 75
6.4.3.1 Modelagem Conceitual de Documentos XML 77
6.4.4 Proposta de Embley 78
6.4.5 O Sistema INFOSLEUTH 78
6.4.6 MOMIS 78
6.4.7 WebKB.....78
7 XML e PHP 80
7.1 A API XML em PHP 80
7.2 Aplicativos XML em PHP 82
8 Descrição do estudo de caso..... 83
8.1 Introdução 83
8.2 Descrição e estrutura dos cursos envolvidos.....84
8.2.1 Curso de informática (ensino superior).....84
8.2.2 Curso técnico de informática (Ensino de nível médio).....85
8.3 Vantagens da disponibilização na Web.....85
8.4 A aplicação.....86
8.5 Justificativa da arquitetura utilizada.....90
9 Conclusões.....99
Bibliografia 101

Lista de Abreviaturas

API	Aplication Program Interface
BDSE	Banco de Dados Semi-Estruturado
BD	Banco de Dados
CDF	Channel Definition Format
CSS	Cascading Style Sheets
DTD	Document Type Definition / Declaration
DHTML	Dynamic HyperText Markup Language
DOM	Document Object Model
DBs	Bancos de Dados
DSSSL	Document Style Semantics ad Specification Language
EDI	Intercâmbio Eletrônico de Dados
FDD	Frames de Descrição de Dados
HTML	HyperText Markup Language
IA	Inteligência Artificial
MML	Mathematical Markup Language
OTP	Open Trading Protocol
OID	Object Identifier
RDF	Resource Description Framework
RI	Recuperação da Informação
SAX	Sample API for XML
SIG	Sistema de Informação Global
SGML	Standard Generalized Markup Language
URIs	Universal Resource Identifier
XSL	eXtensible Style Language
XSLT	XSL Transformations Specification
XML	eXtended Markup Language
W3C	World Wide Web Consortium
WWW	World Wide Web

Lista de Figuras

FIGURA 1- As informações mostradas no Browser *Internet Explorer* 15
FIGURA 2- Documento HTML conectado à World Wide Web..... 15
FIGURA 3- Documento XML conectado à WWW. 16
FIGURA 4- Conteúdo, estrutura e apresentação em documentos na forma tradicional x XML. 17
FIGURA 5- Documento XML contendo um subconjunto externo. 18
FIGURA 6- DTD referenciada na Figura 5 (alunos.dtd). 18
FIGURA 7- Documento XML contendo um subconjunto interno. 18
FIGURA 8- Utilização de elementos e subelementos. 21
FIGURA 9- Utilização de elemento vazio e atributos. 21
FIGURA 10- Formato geral dos comentários em XML. 22
FIGURA 11- Formato geral de uma seção CDATA 23
FIGURA 12- Exemplo de um elemento declarado com dados de caractere 23
FIGURA 13- Conteúdo válido do elemento <TITULO> conforme declaração na DTD. 24
FIGURA 14- Variações do conteúdo do tipo de elemento. 24
FIGURA 15- Exemplo de um documento XML. 26
FIGURA 16- DTD do exemplo da Figura 15. 26
FIGURA 17- Forma geral de uma declaração de lista de atributos. 27
FIGURA 18- XML *Schema* equivalente ao DTD da Fig. 16. 33
FIGURA 19- Exemplo do uso do *namespace*..... 35
FIGURA 20- Parte de um exemplo de uma CSS simples. 37
FIGURA 21- Parte de um documento XML, vinculado a uma folha de Estilo CSS. 37
FIGURA 22- Arquitetura de um programa XML. 42
FIGURA 23- Linguagens suportadas pela XML. 48
FIGURA 24- Modelo XML-GDM para a DTD da Fig.16. 57
FIGURA 25- Consulta XML-GL do tipo *extract-clip*. 58
FIGURA 26- Relacionamento entre as linguagens Lorel e OQL. 61
FIGURA 27- Exemplo de uma consulta simples em XQL. 62
FIGURA 28- A Web de acordo com a linguagem WebSQL. 63
FIGURA 29- Hypertree das publicações de um grupo de pesquisa. 64
FIGURA 30- Arquivo XML da aplicação. 65
FIGURA 31- Grafo XML não ordenado da DTD da Figura 16 67
FIGURA 32- Grafo XML ordenado da DTD da Figura 16 68
FIGURA 33- Ontologia para o domínio da aplicação 72
FIGURA 34- Especificação parcial em lógica de frames da aplicação. 77
FIGURA 35- Resultado de um script em php manipulando o arquivo TEST.XML 81
FIGURA 36- Tela de abertura da aplicação 86
FIGURA 37- Aplicação- OPÇÃO LISTAGEM 87
FIGURA 38- Aplicação- OPÇÃO CONSULTA 87
FIGURA 39- Aplicação- OPÇÃO VISUALIZAÇÃO INDIVIDUAL 88
FIGURA 40- Aplicação- OPÇÃO VISULAIZAÇÃO INDIVIDUAL COMPLETA 88
FIGURA 41- Aplicação- OPÇÃO CURSO TÉCNICO DE INFORMÁTICA 89
FIGURA 42- AplicaçãO- OPÇÃO CURSO DE INFORMÁTICA 89
FIGURA 43- Aplicação – opção DETALHES 90

FIGURA 44- XML NOTEPAD 90
FIGURA 45- Edição do arquivo *alunos.xml* 91
FIGURA 46- Arquivo *tela1.htm* 93
FIGURA 47- Consulta pelo nome do aluno 94
FIGURA 48- Arquivo curso.htm 95
FIGURA 49- Arquivo curso.php 96
FIGURA 50- Arquivo gecur.php 97
FIGURA 51- Arquivo gecur.php (continuação Fig. 50) 98

Lista de Tabelas

TABELA 1. Variações no modelo de conteúdo do elemento 25
TABELA 2. Tipos de Atributos 28
TABELA 3. Comparação entre SQL e XQL 62

Resumo

Nos últimos anos, um grande esforço tem sido despendido no estudo de formas de representar documentos textuais, chamados semi-estruturados, objetivando extrair informações destes documentos com a mesma eficiência com que essas são extraídas de bancos de dados relacionais e orientados a objetos.

A pesquisa, em dados semi-estruturados, tornou-se fundamental com o crescimento da Web como fonte e repositório de dados, uma vez que os métodos de pesquisa existentes, baseados em navegação e busca por palavra-chave, mostraram-se insuficientes para satisfazer as necessidades de consulta em escala cada vez maior.

Com o surgimento da XML, e a tendência de se tornar a linguagem padrão na Web, fez com que a representação de informações fosse dirigida para este novo padrão, porque disponibiliza um bom intercâmbio de informações e a produção de documentos eletrônicos.

Existe a necessidade de se disponibilizar os documentos e as informações na rede em formato Web, HTML ou XML. Sendo assim, consultar documentos XML representa um desafio para a comunidade de pesquisa em banco de dados, pois implica em disponibilizar os grandes volumes de dados já existentes em formato XML, surgindo a necessidade de ferramentas de consulta que sejam ao mesmo tempo, flexíveis o suficiente para compreender a heterogeneidade dos documentos e, poderosas ao ponto de extraírem informações com rapidez e correção.

Este trabalho apresenta uma abordagem sobre a linguagem XML, sua importância, vantagens, principais aplicações e as linguagens de consulta para dados XML. Após, é detalhada uma aplicação para Web utilizando a tecnologia XML, baseado em Ontologias, e a sua disponibilização na Web.

A aplicação desenvolvida utilizou XML e linguagens de consulta para *XML* e com suporte a XML, baseando-se em *Ontologias*, com o objetivo de permitir consultas e o armazenamento de informações referentes aos alunos concluintes de determinados cursos da Universidade da Região da Campanha - Urcamp/Bagé, colocando à disposição da Universidade uma nova ferramenta, que disponibiliza informações referentes aos cursos em questão, utilizando uma nova tecnologia, que tende a tornar-se padrão na Web.

Palavras-chave: World Wide Web, Web, Internet, XML, Extensible Markup Language, Linguagens de Consulta para Web, Aplicações em XML.

TITLE: "XML APLICATION IN ACADEMIC STRUCTURE BUILDING OF ENVIROMENTS BASED ON ONTOLOGIES"

Abstract

At last years, a great effort has been expended in the study of forms to represent textual documents, denominated semi-structurated, objectifying to extract information of these documents with the same efficiency with that these are extracted of data bases related and guided objects.

The research, in semi-structurated data, became fundamental with the growth of the Web as source and repository of data, a time that the existing research methods, based in navigation and it searches for key word, had revealed insufficient to satisfy the necessities of consultation in large scale.

With the sprouting of the XML, and the trend of becoming the standard language in the Web, it made with that representation of information was directed for this new standard, because available a good interchange of information and the electronic document production.

There are necessity of dispose documents and the information at Web, HTML or XML format. Then, to consulting XML documents represent a challenge for community of research in data base, therefore it implies in dispose the great volumes of existing data already in XML format, appearing the necessity of consultation tools that are at the same time, flexible the sufficient to understand the different kind of documents and, powerful to the point to extract information with rapidity and correction.

This work presents a boarding on XML language, its importance, advantages, main applications and the languages of consultation for XML data. After, is detailed an application for Web using technology XML, based on Ontologies, and its dispose in the Web. The developed application used XML and languages of consultation for XML and with support the XML, based on Ontologias, with objective to allow to consultations and the storage of referring information to academic of some courses of the Universidade da Região da Campanha - Urcamp/Bagé, placing to disposal of the University a new instrument, that available information at courses in question, using a new technology, that tends to become standard in Web.

Keywords: World Wide Web, Web, Net, XML, Extensible Markup Language, Languages of Consult for Web, XML Applications.

1 Introdução

A importância da padronização dos dados, principalmente em formato texto, é algo que vem preocupando crescentemente pesquisadores das mais diversas áreas de pesquisa, pelo fato de que a Web vem se tornando um meio de disseminação de informação descontrolada nos últimos anos. A padronização facilita a troca de informações entre aplicativos de todos os tipos.

Segundo [LIG99], [MOU2000], [MAR2000], a razão de se utilizar XML, deve-se ao fato da marcação ser clara, sendo similar a HTML, cujo uso é global e evidente na Web.

Nos últimos anos, um grande esforço tem sido despendido no estudo de formas de representar documentos textuais, chamados semi-estruturados, objetivando extrair informações destes documentos com a mesma eficiência com que essas são extraídas de bancos de dados relacionais e orientados a objetos. A pesquisa em dados semi-estruturados tornou-se fundamental com o crescimento da Web como fonte e repositório de dados, uma vez que os métodos de pesquisa existentes, baseados em navegação e busca por palavra-chave, mostraram-se insuficientes para satisfazer as necessidades de consulta em escala cada vez maior.

Com o surgimento da XML, e a tendência de se tornar a linguagem padrão na Web, fez com que a representação de informações fosse dirigida para este novo padrão, porque disponibiliza um bom intercâmbio de informações e a produção de documentos eletrônicos, [KAD 99].

Existe claramente a necessidade de se disponibilizar os documentos e as informações na rede em formato Web, HTML ou XML. Sendo assim, consultar documentos XML representa um desafio para a comunidade de pesquisa em banco de dados, pois implica em disponibilizar os grandes volumes de dados já existentes em formato XML, surgindo a necessidade de ferramentas de consulta que sejam ao mesmo tempo, flexíveis o suficiente para compreender a heterogeneidade dos documentos e poderosas ao ponto de extraírem informações com rapidez e correção.

Em XML, projetistas podem criar seus próprios elementos de acordo com a aplicação que está sendo modelada, dando importância ao conteúdo e a estrutura da informação, sem se preocupar com a apresentação. Outros benefícios disponíveis no WWW são os editores de documentos XML, permitindo a criação simplificada de documentos em XML.

Além disso, XML é uma linguagem derivada da SGML, porém de mais fácil utilização e aceitação, desenvolvida para troca de informações entre diversas aplicações, plataformas e ambientes na rede, na Internet como em Intranets.

A perspectiva de um padrão útil, fácil e que esteja de acordo com as necessidades de troca de informação e visualização de dados exigidas na Web e em ambientes de integração, tem feito com que pesquisadores do mundo utilizem XML em seus projetos de pesquisa.

A estrutura oferecida, acaba com uma carência dos dados semi-estruturados encontrados, que é a ausência de uma forma de representação de sua estrutura. È claro que está longe de possuir uma estrutura perfeita, como no caso de banco de dados convencionais, mas pode ser considerada uma estrutura *self-describing*, ou seja, auto-descritiva, possui uma estrutura de descrição própria, o que é comentado na seção 2, em maiores detalhes.

XML vem se firmando como padrão para o intercâmbio de informações e produção de documentos eletrônicos, dessa forma o desenvolvimento de aplicações, a integração de dados oriundos de diferentes fontes, linguagens de consulta utilizadas para estes documentos passaram a receber atenção especial por parte dos pesquisadores.

A Ontologia, já é um termo conhecido nas áreas da Filosofia, porém na década de 90 passou a ser empregado em IA, para descrever conceitos e relacionamentos, em BD no qual é considerado um modelo conceitual, que descreve a estrutura do banco de dados em um alto nível de abstração.

Segundo [SIL99], uma Ontologia pode ser definida como um conjunto de conceitos de interesse em determinado domínio de aplicação, juntamente com os relacionamentos entre estes conceitos. Ontologias fornecem uma descrição concisa, uniforme e declarativa das informações semânticas, independente do local e da forma como os dados estão armazenados.

O autor menciona várias abordagens de aplicação de Ontologias sobre dados oriundos da Web, sobre dados textuais ou sobre dados semi-estruturados e sobre sistemas de informações globais.

A partir destas abordagens pode-se aplicar XML, com a constituição de uma camada semântica sobre os repositórios de dados descritos sobre determinada Ontologia, sobre a qual os usuários executam consultas, ficando o sistema com a responsabilidade de gerenciar o acesso aos dados.

O objetivo deste trabalho é apresentar uma abordagem bastante ampla sobre XML, a sua importância e vantagens, após o desenvolvimento de uma aplicação para Web utilizando XML e baseado em Ontologias, após a sua disponibilização na Web.

A aplicação desenvolvida utiliza XML e linguagens de consulta para *XML*, baseado em *Ontologias*, com o objetivo de permitir consultas e o armazenamento de informações referentes aos alunos concluintes de determinados cursos da Universidade da Região da Campanha - URCAMP/Bagé, colocando à disposição da Universidade uma nova ferramenta, que disponibiliza informações referentes aos cursos em questão e utilizando uma nova tecnologia, que tende a se tornar padrão na Web.

Permite a visualização e a manipulação dos dados referentes, aos estágios de conclusão de curso dos alunos dos Cursos de Informática (Ensino Superior/3º grau) e Técnico em Informática (Ensino Médio/2º grau), da URCAMP/Bagé-RS.

As informações manipuladas e armazenadas pela aplicação, aqui descrita, são referentes ao nome do aluno concluinte, tema do trabalho ou estágio desenvolvido para a conclusão do referido curso, um pequeno resumo do trabalho desenvolvido, a nota final do estágio e informações relativas ao professor orientador.

Além disso, a aplicação permite consultas pelo nome dos alunos, por área de estágio, por professor orientador, por ano ou semestre de conclusão do curso.

2 XML - Extensible Markup Language

A importância da padronização de dados, principalmente em formato texto, com o objetivo de facilitar a troca de informações entre diferentes aplicativos, é algo que vem preocupando os pesquisadores nas mais diferentes áreas de pesquisa, pelo fato de que a Web vem se tornando um meio de disseminação de informação descontrolada nos últimos anos.

A utilização da XML– eXtended Markup Language (Linguagem de Marcação Estendida), para padronização destes documentos, é sugerida devido a sua similaridade com HTML, cujo uso é global e evidente na Web. Além disso, XML é uma linguagem para troca informações entre diferentes aplicações, plataformas e ambientes de rede, tanto na Internet como em Intranets, [LIG99] e [MOU00].

XML é resultado de um projeto desenvolvido pelo World Wide Web Consortium – W3C, criado com o objetivo de fornecer a comunidade um nova linguagem para a Web, compatível com SGML e, também, permitir uma implementação rápida e fácil, proporcionando interoperabilidade com HTML e SGML, [W3C99].

Segundo [GRA 99], XML possui uma estrutura flexível, permitindo ao usuário criar suas linguagens de marcação personalizadas, porém todas as linguagens baseadas em XML compartilham a mesma aparência, impressão e sintaxe básica. Além disso, já existe um certo número de linguagens baseadas em XML de padrão industrial em áreas como tecnologia *Push* (CDF- Channel Definition Format), Comércio Eletrônico – e-Commerce (OTP- Open Trading Protocol) e Matemática (MML - Mathematical Markup Language).

Qualquer linguagem baseada em XML é composta de um conjunto de tipos de elementos que servem para definir os tipos de documentos que são conhecidos como Document Type Definitions, ou DTDs.

A principal vantagem de permitir que sejam definidas linguagens próprias, é que permite a liberdade da captura e publicação de informações úteis sobre como os seus dados e estruturas ficarão, ao invés de se acostumar com um formato utilizado por outras pessoas, o que pode ser melhor visualizado através do exemplo abaixo.

Supondo a necessidade de se publicar dados sobre os Estágios de Conclusão de Curso de uma Universidade na Web:

Aluno*:* **Carla Martins Basso**

Informações sobre o estágio*:*

Área*:* **Desenvolvimento para a Internet/XML**

Prof. Orientador*:* **José Palazzo Moreira de Oliveira**

Local*:* **UFRGS/URCAMP**

Para publicar essas informações, usando HTML, seria necessário criar um documento mais ou menos da seguinte forma:

```
<!-- Exemplo em HTML -- >
<h1>Informações sobre estágios de uma Universidade</h1>
<h2>Aluno: Carla Martins Basso </h2>
<table border = 1>
<tr>
    <td>Área:  </td> <td> Desenvolvimento para a Internet/XML </td>
</tr>
<tr>
   td>Prof. Orientador: </td> <td>José Palazzo Moreira de Oliveira</td>
</tr>
<tr>
   <td>Local: </td> <td>UFRGS/URCAMP</td>
</tr>
</table>
```

O documento acima, quando for aberto em um browser ou navegador, as informações seriam mostradas de acordo com a Figura 1, observando-se que, as informações úteis sobre o que realmente elas representam, foram perdidas, tudo o que é informado são níveis de título, tabela, fonte, como mostra a Figura 2.

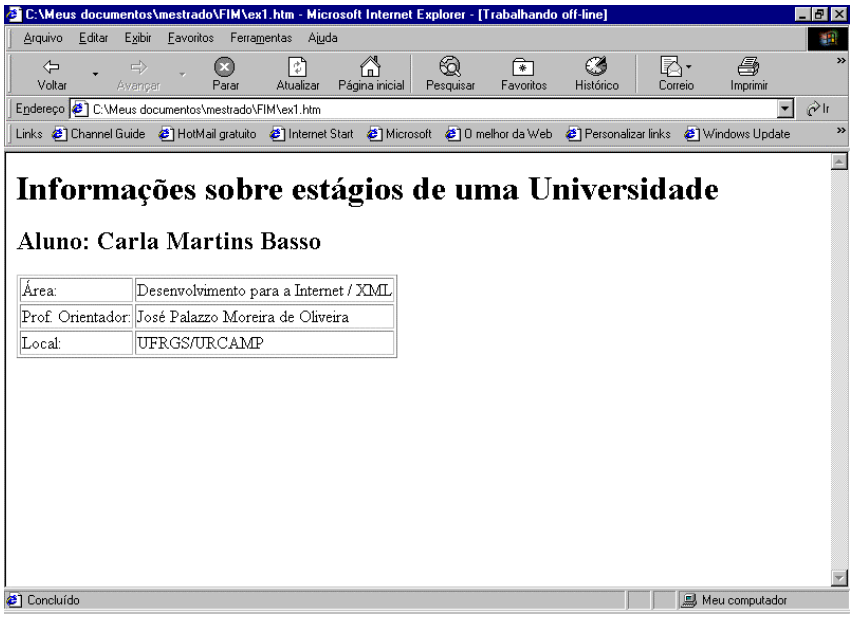

FIGURA 1. As informações mostradas no Browser *Internet Explorer*.

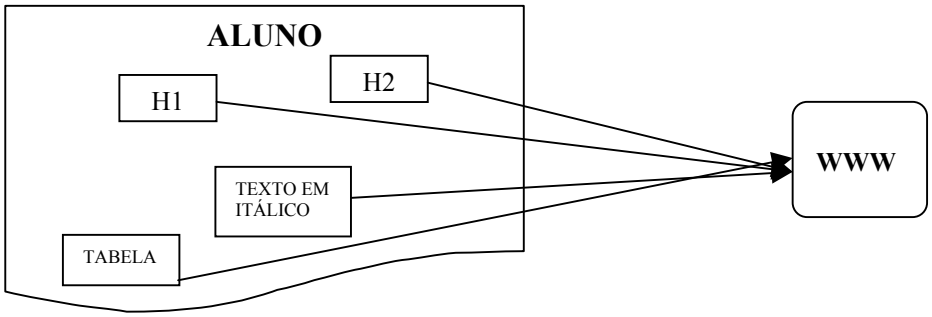

FIGURA 2. Documento HTML conectado à World Wide Web.

Para publicar as mesmas informações na Web, utilizando XML, seria da seguinte forma:

```
<!-- Exemplo em XML -- >
<est>Informações sobre estágios de uma Universidade</est>
<estagios>
        <nome>Carla Martins Basso </nome>
        <area>Desenvolvimento para a Internet/XML</area>
        <orient>José Palazzo Moreira de Oliveira</orient>
        <local>UFRGS/URCAMP</local>
</estagios>
```

Dessa forma, é permitido ao navegador fazer o trabalho de formatação dos dados na tela do usuário, permitindo a execução de cálculos a partir destes dados, que poderão ser exibidos e manipulados de diferentes formas, possibilitando a pesquisa inteligente das informações, realizando pesquisas complexas, criando vínculos inteligentes entre tipos diferentes de informações. A estrutura de um documento XML conectado à WWW pode ser observado na Figura 3.

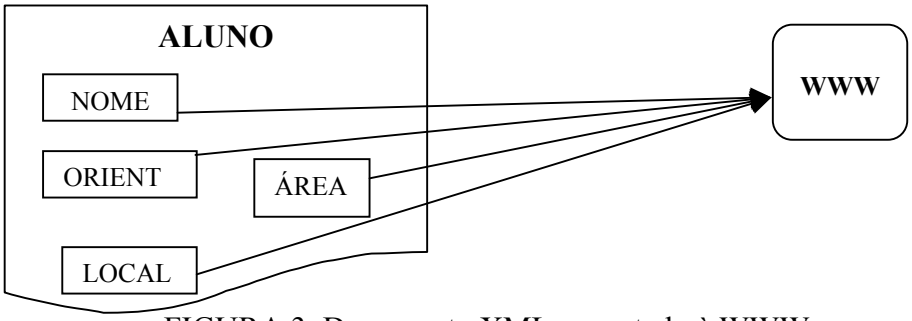

FIGURA 3. Documento XML conectado à WWW.

Segundo [GRA 99], a linguagem XML surgiu como resultado de uma análise longa e cuidadosa do que realmente significa o termo "documento" no mundo digital, que de uma forma geral são compostos de três elementos distintos:

- ***Conteúdo de dados*** – que são as palavras propriamente ditas;
- ***Estrutura*** – corresponde ao tipo de documento e a organização de seus elementos, ou seja a ordem com que eles aparecem;
- ***Apresentação*** – é a forma com que as informações são mostradas ao leitor, ou seja, por voz(som), através da tela de um browser, entre outras.

A idéia central da XML é manter essas características citadas acima, e separadas umas das outras, de forma que se tornem explícitas para o sistema computacional.

Na Figura 4, é exibido as diferenças no tratamento desses três aspectos de um documento, em um processador de texto tradicional e o tratamento dado pela XML no qual é dado atenção especial ao significado das informações e no modo como são estruturados.

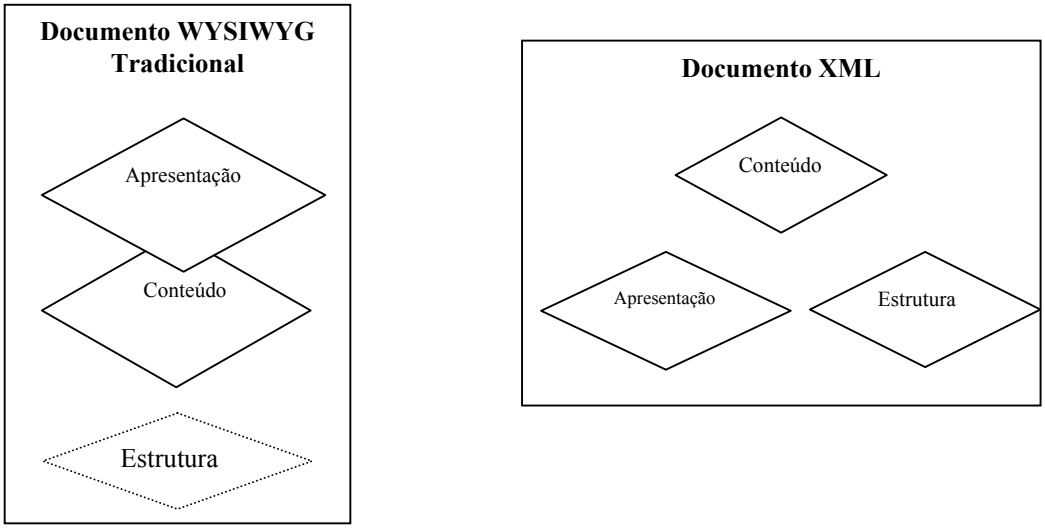

FIGURA 4. Conteúdo, estrutura e apresentação em documentos na forma tradicional comparada à XML.

2.1 Estrutura de um documento XML

Segundo a especificação de XML, [W3C 99], cada documento XML contém uma *estrutura lógica*, composta por *elementos*, e *uma estrutura física*, composta por *entidades, comentários, referências de caracteres e instruções de processamento*, todos estes indicados por marcações explícitas no documento. Os documentos XML são compostos de uma ou mais *entidades*, que são as unidades de armazenamento básicas da XML, podendo conter texto em XML, para ser interpretado pela marcação XML e fornecem a sintaxe para permitir que entidades e elementos sejam declarados e assim possam ser interpretados corretamente por um processador XML.

Basicamente, a estrutura de um documento XML pode ser dividida da seguinte maneira:

a) Prólogo

É a parte de um documento XML, que precede a primeira *tag inicial*, podendo estar completamente vazia, devendo conter pelo menos uma instrução de processamento, a declaração XML.

No prólogo aparecem as informações que identificam o documento como sendo XML, tais como a versão da linguagem e o tipo de documento ao qual ele está associado. Se o documento XML for associado a uma DTD (Document Type Definition), então esta deverá aparecer no prólogo.

A declaração XML mais simples ficará da seguinte forma:

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

b) Document Type Definition (DTD)

A DTD - Document Type Definition (Definição do Tipo de Documento), é o nome dado à gramática geral definida pelos construtores que aparece na DTD (Document Type Declaration), ou seja, o resultado de quando o interpretador processou todas as declarações hospedadas nos subconjuntos interno e externo da DTD.

A DTD representa a área do prólogo usada para declarar, tipos, atributos, entidades; é apresentando a seguinte forma geral:

```
<!DOCTYPE ...>
```

A declaração do tipo de documento não é obrigatória em XML, porém é recomendável utilizá-la, pois garante que o documento XML se encontra em um formato conhecido, facilitando tarefas como a criação de uma folha de estilo para o documento. Assim, documentos que fazem parte de um sistema de informação devem ser validados por uma DTD, o que garantirá a sua consistência, facilitando a programação, tornando os resultados mais confiáveis e a criação de uma folha de estilo para este documento.

A Figura 5 mostra um exemplo de um documento XML com uma declaração do tipo de documento, contendo um subconjunto externo (Figura 6), através de uma referência a uma entidade externa., representada pela palavra *doctype*. Porém, também pode ocorrer um subconjunto interno, conforme é ilustrado na Figura 7.

```
<? xml version = "1.0" ?>
<!DOCTYPE alunos SYSTEM "alunos.dtd">
<greeting> Hello, world! </greeting>
```

FIGURA 5. Documento XML contendo um subconjunto externo.

```
<!ELEMENT nome (#PCDATA)>
```

FIGURA 6. DTD referenciada na Figura 5 (alunos.dtd).

```
<!DOCTYPE alunos [
>!ELEMENT nome [#PCDATA]>
]>
<nome>João da Silva</nome>
```

FIGURA 7. Documento XML contendo um subconjunto interno.

c) Elemento Raiz

É o elemento que contém todos os outros elementos no documento, podendo também estar vazio, porém só pode ocorrer um único elemento raiz, em um documento XML. No exemplo abaixo elemento raiz é representado por *book*:

```
<?xml version = "1.0"?>
<book>
  .......os demais elementos aparecem aqui....
</book>
```

d) Epílogo

É usada para dar nome a área que pode conter instruções de processamento, comentários ou espaços em branco, embora não seja oficialmente usada no padrão XML.

2.2 As principais diferenças entre XML e HTML

XML é um veículo excelente para se publicar dados na Web, que devido as habilidades referentes a apresentação, do HTML, juntas as duas linguagens de marcação se complementam. Diferentemente do HTML, XML é ideal para a criação de aplicativos Web complexos, principalmente quando as opções de apresentação mais simples do HTML forem ultrapassadas.

A maioria dos browsers utilizados atualmente, apresenta suporte ao XML, com a capacidade de vinculação de dados do DHTML e do CDF, e também trazem suporte ao DOM - Document Object Model (modelo de objetos de documento), ou seja, os elementos utilizados podem ser definidos em XML na forma de objetos, que poderão ser programados através de linguagens script, tornando assim, XML a linguagem ideal para exibir dados dinâmicos em plataformas diferentes.

Além de permitir especificar dinamicamente que os dados se modificam em seus documentos, XML torna possível criar fontes de dados que podem ser usadas em diversas finalidades que, a interoperabilidade entre aplicativos de um servidor ou em uma máquina cliente e seu site [MOU00].

Em termos básicos XML e HTML diferem em três aspectos fundamentais:

- XML permite definir o seu próprio conjunto de nomes de atributos e marcas, toda a vez que se fizer necessário, permitindo que o usuário crie novas marcas para indicar a estrutura, não oferecendo nenhuma instrução de como o conteúdo deverá ser exibido, esta informação é incluída separadamente em uma folha de estilo, que traduz os dados XML para HTML e, que após são exibidas no browser;
- as estruturas de dados, em XML, podem ser aninhadas em qualquer nível de complexidade, desde que sigam as regras da linguagem;
- os documentos XML podem conter uma descrição opcional da gramática do documento de modo que outros aplicativos que necessitem executar processos de validação na estrutura dos documentos possam assim fazê-lo.

Segundo [ABI00], XML em sua forma básica, é simplesmente uma sintaxe para transmissão de dados, devendo se tornar o formato universal para troca de dados na Web, apresentando uma especificação que facilita esta troca de dados e a reutilização por múltiplas aplicações, surgindo assim ferramentas para importação e exportação de dados em XML.

2.3 Sintaxe Básica da Linguagem XML

2.3.1 Elementos

XML é uma representação textual dos dados cujos componentes básicos são chamados de *elementos*, que representam os componentes lógicos do documento. Os elementos podem conter outros elementos, além de palavras e frases, que constituem o texto propriamente dito, denominado dados de caracter do documento. A hierarquia dos componentes constitui a árvore deste documento.

O elemento que contém todos os outros elementos, é chamado de *elemento raiz* (*root element*) e os elementos nele contidos são os *subelementos*, que por sua vez, podem conter outros *subelementos* ou apenas dados de caracter. O elemento raiz também pode ser chamado de *elemento documento* (*document element*), pois armazena toda a lógica do documento.

A maioria dos elementos em XML apresentam conteúdo. Elementos com conteúdo são representados por uma *tag de início* (*<nome_elemento>*), o conteúdo do elemento e, uma *tag de finalização* (*</nome_elemento>*).

Elementos que não apresentam conteúdo são denominados *elementos vazios*, podendo apresentar uma tag de início e una tag de finalização sem nada entre elas (*<book></book>*), ou uma única tag de elemento vazio (*</book>*).

2.3.2 Atributos

Os elementos podem apresentar informações anexadas extras chamadas de *atributos*, que descrevem as propriedades do elemento. Em XML, atributos são definidos como pares (nome, valor), como no exemplo abaixo, onde o atributo, usado para especificar a linguagem corrente, a moeda e o formato do endereço, aparece sempre entre aspas.

```
<produto>
        <nome linguagem=”Português”> Iniciando em XML </nome>
        <moeda corrente=”real”>120,00 </moeda>
        <endereço formato=”padrão” linguagem=”português”>
        <rua> Nei Brasil Lemos </rua>
        <cep> 96415-520 </cep>
        <cidade> Bagé </cidade>
        </endereço>
</produto>
```

A lista de atributos de cada elemento é definida na DTD do documento e, usualmente, elementos vazios apresentam atributos. XML apresenta diversas formas de representar a mesma informação, o que justifica o uso de DTDs, que estabelecem estas restrições, mantendo a consistência entre os documentos.

A Figura 8 mostra um exemplo utilizando *elementos* e *subelementos*, enquanto que a Figura 9 utiliza elemento vazio e atributos.

```
<remetente>
<nome>Carla Basso</nome>
<email>”carla@alternet.com.br”</email>
</remetente>
```

FIGURA 8. Utilização de elementos e subelementos.

```
<remetente nome = ”Carla Basso”   email = ”carla@alternet.com.br”/>
```

FIGURA 9. Utilização de elemento vazio e atributos.

2.3.3 Entidades

Um documento XML é definido como uma seqüência de caracteres, percorridos linearmente pelo processador XML. Para permitir a criação de documentos não lineares e em múltiplos pares, existe um mecanismo, utilizando *entidades*, que permite que o *parser* XML reorganize estes pedaços em uma estrutura linear.

As entidades podem ser constituídas por um único caracter, ou por todos os caracteres de um documento, sendo também possível, inserir no documento uma referência a uma entidade através do nome da entidade, sendo assim o processador XML, substitui a referência da entidade pela entidade propriamente dita.

A utilização de entidades visa prover um mecanismo de gerenciamento de tamanho e complexidade de documentos, permitindo a reutilização de textos em diferentes contextos automaticamente, proporcionando aos programadores um melhor gerenciamento dos recursos disponíveis como memória, velocidade de conexão, entre outras.

XML dispõe de uma série de técnicas para manipulação de dados, que não apresentam marcação XML, tais como imagens, sons, filmes, ou qualquer outro tipo de dados que não seja XML (incluindo HTML e outras formas de SGML), que através de uma notação associada que declara o tipo da entidade, ou seja, se é um *GIF, JPEG, MPEG*, permite o seu tratamento adequado.

A última função das entidades é representar quaisquer caracteres *Unicode*, o que permite a produção de documentos em qualquer idioma. As entidades sempre começam com e-comercial (&) e terminam com ponto e vírgula(;).

2.3.4 Caracteres de dados e Marcação

Os caracteres em um documento XML representam uma combinação de *caracteres de dados*, que é o verdadeiro texto do documento, e a *marcação*, a informação que adiciona valor ao texto [LIG 99].

A marcação (*markup*) representa a estrutura lógica e conecta as entidades. Um documento XML é composto exclusivamente por marcação e dados de caracter, que combinados, representam o texto XML.

Construções como tags, referência a entidades, declarações, são exemplos de marcação, e constituem a parte do documento compreensível pelo processador XML. Os dados de caracter, contidos na marcação, constituem a parte do documento que deverá ser interpretada pelo usuário e não pelo processador.

Toda a marcação inicia pelos símbolos menor (<), maior(>), ou &, e qualquer outro dado é considerado dado de caracter.

2.3.5 Manipulação de espaços em branco

Em XML, espaços em branco, referem-se a um ou mais caracteres de espaço, tabulação, *carriage return* ou alimentação de linha. Em geral, é interessante o uso de espaços em branco para mostrar e estrutura hierárquica do documento, porém o seu uso representa um problema para o processador XML, ou seja, como diferenciar quais os espaços em incluídos para fins de layout, e quais que realmente fazem parte do documento.

A regra geral para o elemento raiz é que qualquer espaço em branco que ocorrer fora de uma construção de marcação, tal como uma tag inicial, é considerado significativo e é passado para o aplicativo. O espaço em branco não é significativo no prólogo e no epílogo.

2.3.6 Comentários

Os comentários apresentam a forma geral apresentada na Figura 10. Não é permitido a ocorrência de uma string "--" dentro de um comentário, porém com exceção disso, tudo é permitido, inclusive novas linhas. Os comentários podem aparecer em qualquer lugar do documento XML, pois não são considerados como parte da estrutura lógica do documento.

```
<!—aqui aparece o comentário desejado -->
```

FIGURA 10. Formato geral dos comentários em XML.

2.3.7 Seções CDATA

Uma seção CDATA é usada pata proteger o corpo do texto das atenções do processador XML, informando ao *parser*[1], que ele deve ignorar os caracteres de marcação. Uma vez na seção CDATA, a única marcação a que o processador XML dará atenção em especial, será a string "]]>"que termina a seção CDATA.

O formato geral de uma seção CDATA é mostrado na Figura 11, na qual estão em destaque os caracteres usados para iniciar e terminar uma seção CDATA. Muitas vezes é necessário que um documento contenha um grande número de caracteres considerados especiais pelo processador XML, como por exemplo, "&" e "<", para isolar este bloco de texto, utiliza-se uma seção CDATA.

<[CDATA [if a>b and b<c them A<c **]]>**

FIGURA 11. Formato geral de uma seção CDATA.

2.3.8 Instruções de Processamento

As instruções de processamento (*Processing Instructions – PIs*) são seções que não fazem parte do documento, mas que devem ser repassadas para alguma aplicação pelo processador XML. As *PIs* apresentam o formato **<?nome, dados?>**, onde o nome, chamado de *PI target*, indica a *PI* para a aplicação, que caso reconheça, utiliza os dados que a seguem. Os nomes das PIs que começam com XML são reservados para a padronização da linguagem.

2.3.9 Dados de Caracteres

Alguns elementos podem conter apenas dados de caracteres, ou texto, e nenhum outro elemento adicional, sendo este o modelo mais simples de conteúdo que um elemento pode conter. Nenhum outro elemento pode ser aninhado, pois se o elemento for declarado como dados de caracteres o único conteúdo que poderá ser incluído entre as tags de abertura e fechamento é texto puro, Figura 12.

<!ELEMENT TITULO (#PCDATA)>

FIGURA 12. Exemplo de um elemento declarado com dados de caractere.

[1] Significa interpretar, tarefa fundamental de um processador XML, e refere-se ao processo por meio do qual os caracteres que formam o documento XML são categorizados tanto como marcação ou dados de caractere.

Uma vez declarado o elemento TITULO (Figura 12), esse elemento poderia ser colocado no documento XML com o conteúdo mostrado na Figura 13 que seria perfeitamente válido e, nenhum outro elemento poderá ser colocado dentro do elemento <TITULO>, o que levaria o documento a ser considerado como inválido pelo *parser*. Não existe limite para a quantidade de texto que poderá ser incluída no elemento declarado como dados de caracter.

```
<TITULO>Bem vindo ao mundo do XML</TITULO>
```

FIGURA 13. Conteúdo válido do elemento <TITULO> conforme declaração na DTD.

2.3.10 Declarações

a) Declarações de tipo de elemento

As declarações de tipo de elemento permitem a um aplicativo XML restringir os elementos que possam ocorrer no documento e especificar a ordem na qual eles ocorrem.

A forma geral de uma declaração de tipo de elemento é a seguinte:

<!ELEMENT....>

A reticência é o espaço reservado para especificação do conteúdo do tipo do elemento, e as três variações possíveis são mostradas na Figura 14.

<!ELEMENT *fruta* **EMPTY>** → declara que os elementos *fruta* são elementos vazios;

<!ELEMENT *fruta* **ANY>** → declara que um elemento *fruta* pode conter qualquer combinação de dados de caractere e outros elementos, desde que esses tipos de elementos tenham sido declarados na DTD;

<!ELEMENT *fruta* (maçã|*laranja*|*banana*)**>** → declara que o elemento *fruta* poderá conter exatamente um elemento *maçã*, *laranja* ou *banana*. A parte entre parênteses é conhecida como *modelo de conteúdo do elemento;*

FIGURA 14. Variações do conteúdo do tipo de elemento.

Variações no modelo de conteúdo do elemento

A XML permite que tipos de elementos sejam declarados para conterem outros elementos, sua ordem e ocorrência podem ser restritas de acordo com as várias formas descritas na Tabela 1.

Tabela 1. Variações no modelo de conteúdo do elemento.

Modelo de Conteúdo	Interpretação
<!ELEMENT aluno (nome,end,telef)>	Um elemento aluno é composto por um elemento nome, um elemento end e um elemento telef, exatamente nesta ordem;
<!ELEMENT fruta (maçã|laranja|banana)>	Um elemento fruta é composto de um elemento maçã, ou um elemento laranja ou um elemento banana, mas apenas um.
<!ELEMENT aluno (nome,end,telef?)>	Um elemento aluno é composto por um elemento nome, seguido de um elemento end e, opcionalmente, por elemento telef.
<!ELEMENT aluno (nome,end,telef*)>	Um elemento aluno é composto por um elemento nome, seguido de um elemento end e por nenhum ou vários elementos telef.
<!ELEMENT aluno (nome,end,telef+)>	Um elemento aluno é composto por um elemento nome, seguido de um elemento end e por um ou mais elementos telef.
<!ELEMENT aluno ANY>	Um elemento aluno é composto de qualquer combinação de elementos em qualquer ordem, desde que tenham sido declarados.

Na Figura 15, é mostrado um exemplo de documento XML, onde podem ser observados elementos, atributos e comentários, já a Figura 16 mostra o DTD para o documento da figura anterior, onde a primeira linha estabelece que o elemento *book*, pode conter um ou mais elementos do tipo *author*, seguido pelo elemento *title* e por um *publisher*.

```
<!—exemplo de um documento XML
aqui começa o prólogo-->
<?xml version=”1.0”>
<!DOCTYPE BOOK SYSTEM “book.dtd”>
<!—aqui começa o elemento raiz -->
<biblio>
<book year=”1999” publisher=”350” >
        <author>
                <firstname>João</firstname>
                <lastname>Silveira</lastname>
        </author>
        <author>
                <firstname>José</firstname>
                <lastname>Macedo</lastname>
        </author>
        <title>XML – Aplicações Práticas</title>
        <ISBN>10-85010-37-8</ISBN>
</book>
<publisher id=”123”>
        <name>Ed.Campus</name>
        <address>Rio de Janeiro,Brazil</address>
<publisher>
<book year=”1998” publisher=””350”>
        <author>
                <firstname>Maria</firstname>
                <lastname>Santos</lastname>
        </author>
        <title>Aplicações na Internet</title>
</book>
</biblio>
```

FIGURA15. Exemplo de um documento XML.

```
<!ELEMENT book (author+,title,ISBN?)>
<!ATTLIST book year CDATA #required>
<!ATTLIST book publisher IDREF #required>
<!ELEMENT author (firstname,lastname)>
<!ELEMENT article (author+,title,year?,(shortversion  longversion))>
<!ATTLIST  article CDATA>
<!ELEMENT title (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT publisher (name,address)>
<!ATTLIST publisher id ID #required>
<!ELEMENT firstname (#PCDATA)
<!ELEMENT lastname (#PCDATA)
<!ELEMENT name (#PCDATA)
<!ELEMENT address (#PCDATA)
<!ELEMENT ISBN (#PCDATA)
```

FIGURA 16. DTD do exemplo da Figura 15.

b) Declarações de lista de atributos

As declarações de lista de atributos servem para especificar o nome, tipo e opcionalmente o valor *default* dos atributos associados a um elemento. A forma geral de uma declaração de lista de atributos pode ser observada na Figura 17.

```
<!ATTLIST element nome_do_atributo (A|B|C)
```

FIGURA 17. Forma geral de uma declaração de lista de Atributos.

Observando a Figura 16, com relação a declaração de atributos, o elemento *book* possui um atributo chamado *year*, enquanto que o elemento *article* possui um elemento chamado *type*, e esse atributo pode receber valores do tipo CDATA, ou seja, qualquer seqüência de caracteres. O atributo *#required* indica que é necessário lhe atribuir algum valor. Os tipos de atributos existentes, são mostrados na Tab. 2.2.

Uma declaração de lista de atributos inclui informações sobre a necessidade ou não de ser fornecido um valor para esses atributos e, caso não exista necessidade, o que o processador XML deve fazer. Existem, entretanto, três variações para os *defaults* dos atributos:

a) **Atributos requeridos** – aqueles que tem que ser fornecido um valor; Ex.: <!ATTLIST *fruta* type CDATA #REQUIRED>;
b) **Atributos implicados** – o processador XML informa que não foi fornecido nenhum valor, decidindo qual o melhor caminho a ser seguido; Ex.: <!ATTLIST *fruta* type CDATA #IMPLIED>;
c) **Atributos fixos** – é fornecido um valor na declaração, não sendo necessário fornecer um valor no documento.
   Ex.: <!ATTLIST *fruta* EDIBLE CDATA #FIXED "YES">.

Tabela 2. Tipos de Atributos.

Atributos	Funções e Exemplos
Atributo de string	Contem um conjunto arbitrário de caracteres de qualquer comprimento, desde que qualquer ocorrência de "<"e "&"sejam ignoradas com referência a entidades. Ex.: <!ATTLIST *foo bar* CDATA...>
Atributo enumerado	Recebe um conjunto fixo de valores fornecidos como parte de sua declaração. Ex.: <!ATTLIST *apple quality* (good|bad|indiferent)...>
Atributo ID	Esses atributos ID que ocorrerem são exclusivos, devendo começar com uma letra, "_"ou ":", embora os dois pontos deva ser evitado, acrescentado-se então o atributo **UniqueName** ao elemento**.** Ex.: <!ATTLIST *foo UniqueName ID.....>*
Atributo IDREF	Os valores atribuídos aos atributos de IDREF devem corresponder ao valor atribuído a um atributo de ID em algum lugar do documento Ex.: <!ATTLIST *bar* References IDREF...>
Atributo IDREFS	Variação na qual é permitido que um atributo contenha várias IDS referenciadas. Ex.: <!ATTLIST *bar* References IDREF...> <bar References = "A1234 R2345"
Atributos ENTITY/ ENTITIES	São tratados em conjunto, o ENTITY deve ter um valor correspondente ao nome de uma entidade não interpretada declarada em algum lugar da declaração de tipo de documento, ENTITIES permite a um atributo conter um ou vários nomes de entidade em seu valor. Ex.: <!ENTITY *bob* SYSTEM "bob.gif"NDATA gif> <!ATTLIST *letter* salutation ENTITY..> o elemento *letter* ficaria assim: <*letter* salutation="bob">
Atributos NMTOKEN/ NMTOKENS	NMTOKEN está restrito a conter caracteres permitidos em um nome: letras, dígitos e qualquer caracter de pontuação, porém não permite caracteres em branco. NMTOKENS pode conter um ou mais NMTOKENS separados por espaços em branco. Ex.: <!ATTLIST product code NMTOKEN....> E os elementos de product ficariam assim: <product code= "Alpha-123"> <product code="333"> Ex. de NMTOKEN inválido: <product code=" A 123">
Atributo NOTATION	Especifica uma ou mais notações declaradas em algum lugar na declaração de tipo de documento, seu valor é restrito a corresponder a uma dessas anotações. Ex.: <!NOTATION EDIFACT SYSTEM "EDIFACT Format"> <!ATTLIST invoice format NOTATION (EDIFACT)> ..... <invoice format = "EDIFACT">

c) Declarações de Entidades

As declarações de entidade permitem que se associe um nome a qualquer fragmento de um documento XML, que pode conter um texto, uma parte de uma DTD ou uma referência a um arquivo externo, como uma figura qualquer ou qualquer outro tipo de dado binário. Existem três tipos de entidades:

a) **interna**: é aquela cujo valor de substituição é armazenado na declaração, permitindo a definição que qualquer tipo de abreviatura, que será, durante o processamento substituída pelo texto informado. Por exemplo se for definida uma entidade <!ENTITY URC "Universidade da Região da Campanha">, pode-se utilizar &URC em qualquer parte do documento que será sempre substituído pelo conteúdo definido;
b) **externa**: permitem a um documento XML fazer referência a arquivos externos, sejam textos ou binários. Se o arquivo for do tipo texto, seu conteúdo será inserido no lugar da entidade e, no caso de arquivos binários é apenas passada a referência para a aplicação que irá ler o documento, cabendo a ela decidir a forma de apresentação do conteúdo;
c) **de parâmetro**: permitem especificar um tipo genérico de entidade que pode, depois servir como base para outras entidades. Por exemplo, se for definida uma entidade chamada *dadosemp,* contendo o nome da empresa, endereço e CGC, pode-se especificar um endereço comercial e um endereço para cobrança, utilizando-se o endereço anteriormente definido.

d) Declarações de Notação

São utilizadas para identificar os tipos de arquivos binários, ou em um formato não XML. Um exemplo seria a declaração de uma notação para identificar um tipo de arquivo como GIF:

<!NOTATION URCAMP12 SYSTEM "GIF">.

2.3.11 Documentos Válidos e Bem-Formados

Segundo [GRA 99], **documentos XML válidos** são documentos que são bem formados e, adicionalmente, correspondem as assim chamadas, *restrições de validade,* na especificação XML. As restrições de validade detalham como um documento XML deve comportar-se no que diz respeito as regras estruturais especificadas na DTD., ou seja, se for declarado no prólogo do documento uma DTD e após, ao escrever o resto do documento, se obedecer todas as regras que ele prescreve, se terá um *documento XML válido.*

Já os **documentos XML bem formados** são documentos que correspondem a todas as assim chamadas *restrições de boa formação* na

especificação da XML, os quais são perfeitos estruturalmente, mas não necessariamente válidos. Todos os elementos são cuidadosamente posicionados, e todas as entidades referenciadas pelo documento são formalmente declaradas, porém nenhuma verificação é feita se estrutura lógica do documento acompanha as regras dadas na DTD.

A recomendação XML do W3C é bastante específica nas regras para que um documento XML seja considerado bem formado, utilizando para tanto, representações formais. A especificação é explícita em proibir que parsers XML tentem corrigir e interpretar documentos que não sejam bem formados (técnica largamente utilizada pelos browsers HTML, tornando-os bastante complexos. Além das regras de marcação, um documento bem formado deve respeitar o seguinte:

- se presente, a instrução de processamento declaração XML, deve iniciar o documento, não sendo admitido espaços em branco anteriores, podendo indicar a versão de *character set* utilizada;
- todo o documento deve ter um elemento raiz que inclui todos os demais;
- é comum este elemento incluir outros elementos, formando a estrutura de aninhamento de elementos.

2.4 Classes de Documentos XML

Os documentos XML podem ser agrupados a fim de constituir classes, as quais um documento pode pertencer. A classe do documento é utilizada para verificar a validação de um documento XML, ou seja, os documentos XML que estão de acordo com uma aplicação em particular formarão as classes. Essas classes poderão ser agrupamentos rigidamente controlados com todos os possíveis documentos em conformidade com uma única DTD, ou então agrupamentos soltos ao redor de uma família de DTDs relacionadas. Em ambos os casos as classes de documentos trarão vantagens.

Quando uma classe de documento é rigidamente controlada por uma única DTD, o software de aplicação pode tratar as regras expressas naquela DTD em vez de tratá-las como um esquema de banco de dados, significando que todos os documentos na classe podem estar sujeitos a um processamento bem específico. A aplicação deve ser armada completamente uma única vez, quando se desenvolve as folhas de estilo e protocolos de links, escrevendo-se assim um software personalizado e, todo o documento associado ou em conformidade com àquela DTD, podendo-se beneficiar de sua aplicação associada.

Existem duas formas de se definir classes de documentos XML: a *Document Type Definition* (DTD) e o *XML Schema*.

2.4.1 Document Type Definition (DTD)

O conceito de DTD foi herdado da SGML, sendo a forma mais comum de definir classes para documentos XML, funcionando como uma gramática, onde são definidos todos os tipos de marcação que poderão ser utilizados, permitindo também, que sejam incluídas restrições aos documentos XML, que poderão posteriormente serem verificadas pelo processador, a fim de determinar ou não a validade do documento. A indicação de qual DTD está ou será utilizada aparece no início do documento na declaração <!DOCTYPE....., o que pode ser observado na Figura 15, onde aparecem bem claramente as declarações de elemento, de atributo, de entidades e notações.

Entre as principais vantagens do uso de DTDs podemos citar:

a) grande disponibilidade de ferramentas que a suportam, tais como processadores SGML e boa parte dos processadores XML;
b) grande número de tipos de documentos já definidos, tais como, HTML, XHTML, DocBook, TEI, J2008, CALS, etc;
c) conhecimento bem difundido e muitos anos de prática.

Como desvantagens podemos citar alguns problemas e deficiências citados por [KAD 99], como :

a) não é escrita em XML;
b) não possui suporte para *namespaces*;
c) oferece recursos muito pobres para se definir os tipos de dados, principalmente para números, datas e valores de moeda;
d) possui um mecanismo de extensão complexo e frágil, baseado em pouco mais que a substituição de caracteres.

2.4.2 XML Schema

Com o objetivo de superar as deficiências citadas no uso das DTDs, o W3C propôs o *XML Schema* [W3C00a], para servir como ferramenta de definição de tipos de documentos XML. O nome *Schema* (esquema), origina-se da área de banco de dados, onde é utilizado para definir a estrutura das tabelas que compõem uma aplicação. A Figura 18 apresenta o XML *Schema* equivalente a DTD da Figura 16.

A sintaxe utilizada pelo XML Schema é totalmente diferente daquela utilizada na DTD, iniciando pelo fato que os esquemas são documentos XML, possuindo elementos e atributos que são definidos por uma DTD. A base da XML *Schema* é a definição de elementos e atributos. O exemplo abaixo mostra o esquema do elemento *title,* definido na DTD da Figura 16.

O conteúdo do *elementType* define o conteúdo válido de um elemento daquele tipo, que neste caso é <mixed/>, significando que o elemento pode conter qualquer combinação de caracteres, e se algum outro elemento for definido dentro de elementType, significa que o mesmo pode estar aninhado dentro dele.

```
<elementType name= “title”>
      <mixed/>
</elementType>
```

A sintaxe para definição de tipos é bastante poderosa, como pode ser observado no exemplo abaixo, onde é definida um tipo de dados “CEP”:

```
<datatype name’”CEP”>
      <basetype name= “string”/>
      <lexicalRepresentation>
              <lexical>99999-999</lexical>
      </lexicalRepresentation>
</datatype>
```

Uma vez definido o tipo de dados, pode-se declarar um elemento como sendo daquele tipo, o que é mostrado abaixo:

```
<elementType name= “CEPAutor”>
    <datatypeRef name= “CEP”/>
</elementType>
```

```
<elementType name= “book”>
    <sequence>
        <elementTypeRef name=  “author” minoccur= “1 “>
        <elementTypeRef name=  “title” minoccur= “1 “  maxoccur=”1”>
        <elementTypeRef name=  “publisher” minoccur= “1 “ maxoccur= “1”>
    </sequence>
</elemenTtype>
<elementType name= “author”>
    <sequence>
        <elementTypeRef name=  “firstname” minoccur= “1 “>
        <elementTypeRef name=  “lastname” minoccur= “1 “  maxoccur=”1”>
    </sequence>
</elementType>
<elementType name= “article”>
    <sequence>
        <elementTypeRef name=  “author” minoccur= “1 “>
        <elementTypeRef name=  “title” minoccur= “1 “  maxoccur=”1”>
        <elementTypeRef name=  “year” minoccur= “1 “ maxoccur= “1”>
        <elementTypeRef name=  “eachArticleType” minoccur= “1 “ maxoccur= “1”>
    </sequence>
</elementType>
<elementType name= “publisher”>
    <sequence>
        <elementTypeRef name=  “name” minoccur= “1 “ maxoccur=”1”>
        <elementTypeRef name=  “address” minoccur= “1 “  maxoccur=”1”>
    </sequence>
</elementType
<elementType name=”firstname”>
    <mixed/>
</elementType>
<elementType name=”lastname”>
    <mixed/>
</elementType>
<elementType name=”title”>
    <mixed/>
</elementType>
<elementType name=”year”>
    <dataTypeRef name=”dataTime”/>
</elementType>
<elementType name=”name”>
    <mixed/>
</elementType>
<elementType name=”address”>
    <mixed/>
</elementType>
<elementType name=”eachArticleType”>
    <datatypeRef name= ärticleType”/>
</elementType>
<datatype name=”articleType”>
    <enumeration>
        <literal>shortversion</literal> <literal>longversion</leteral>
    </enumeration>
</datatype>
```

FIGURA 18. XML *Schema* equivalente ao DTD da Figura 16.

Entre as principais características de *XML Schema*, pode-se citar algumas:

- Utilização de um maior número de tipos de dados, como booleanos, números, datas, horas, URIs, inteiros, números decimais, números reais, intervalos de tempo, entre outros;
- apresenta recursos para criar outros tipos de dados agregados;
- os tipos são definidos pelo usuário, chamados de arquétipos, podendo-se por exemplo definir um tipo chamado "Telefone" e a partir dele definir dois elementos, "TelefoneResid" e "TelefoneComer", como sendo do mesmo tipo;
- agrupamentos de atributos pode ser aplicados a todos os elementos;
- os *arquétipos* são refináveis, ou permitem "herança", o que é considerado uma das características mais significativas;
- enquanto que a DTD é considerada de conteúdo "fechado", pois descreve exatamente o que pode aparecer no conteúdo do elemento, XML Schema apresenta duas outras possibilidades: *open* e *refinable*. No tipo *open*, todos os elementos requeridos devem estar presentes, embora não seja considerado um erro se elementos adicionais forem inseridos. Já no *refinable*, é admitido que elementos adicionais estejam presentes, desde que no esquema seja definido o que eles são;
- suporta *namespaces*, o que torna a validação dos documentos muito mais difícil, e até que o XML Schema não esteja completo, não é recomentdado validar documentos que utilizam *namespaces*. É permitido a combinação de esquemas múltiplos de *namespaces* para validar os documentos, pois é definido mecanismos para composição de esquemas.

2.5 Namespaces

Uma das características de XML é permitir a criação a definição do documento de acordo com o tipo de dados que se deseja armazenar, o que pode trazer problemas quando uma mesma palavra for usada em contextos diferentes. Por exemplo, no caso de haver dois documentos XML, um sobre informações bibliográficas, e outro tratando dos autores, e se tanto os livros, quantos os autores possuírem um atributo "*title*" pode haver confusão no momento da interação desses documentos. Com esse propósito foi criado o conceito de *namespace* em XML (espaços de nomes), que consiste de um prefixo adicionado ao elemento XML, permitindo assim identificar a qual contexto ele está associado.

Pode-se observar no exemplo a seguir, Figura 19, que a definição do *namespace* aparece logo no início do documento e, para cada elemento , é adicionado um prefixo com o nome do namespace, seguido pelo sinal de dois pontos. Assim sendo necessário fazer a interação de um documento com outro, não existe a possibilidade de haver confusão entre o título do livro com o título da pessoa, pois ambos os elementos poderão ser representados pelo elemento *<title>*.

```
<!--aqui começa o prólogo-->
<?xml version="1.0"?>
<DOCTYPE BOOK SYSTEM "book.dtd">
<!--aqui começa a instância-->
<bib:biblio>
   <bib:book year="2000" publisher="345">
   <bib:author>
      <bib:firstname>Alex</bib:firstname>
      <bib:lastname>Ceponkus</bib:lastname>
   </bib:author>
   <bib:author>
      <bib:firstname>Faraz</bib:firstname>
      <bib:lastname>Hoodbhoy</bib:lastname>
   </bib:author>
   <bib:title>Applied XML</bib:title>
   <bib:ISBN>10-4567-67-9</bib:ISBN>
   </bib:book>
<bib:publisher id="345">
   <bib:name>Ed. Jones</bib:name>
   <bib:address>Bagé, RS, Brasil</bib:address>
</bib:publisher>
<bib:book year="1999" publisher="345">
   <bib:author>
      <bib:firstname>Ellionete</bib:firstname>
      <bib:lastname>Harold</bib:lastname>
   </bib:author>
   <bib:title>XML Bible</bib:title>
</bib:book>
</bib:biblio>
```

FIGURA 19. Exemplo do uso do *namespace*.

2.6 Apresentação de um documento XML

Um browser genérico não possui informações suficientes que o permita interpretar um documento XML e produzir uma apresentação amigável ao usuário, pois possui elementos próprios, concebidos pelo autor do documento ou por algum grupo comum de usuários.

As linguagens de ***folha de estilo*** (***style sheets***) permitem que sejam realizadas descrições de estilo de apresentação dos documentos, mantidas separadas da representação do conteúdo dos documentos. Um documento pode então, ser reaproveitado em diferentes aplicações que exigem diferentes visões do mesmo documento. O princípio das linguagens de folha de estilo está em prover uma sintaxe para identificar partes específicas do conteúdo de documentos e um conjunto de estilos ou ações que devem ser realizadas sobre as partes identificadas.

A informação da folha de estilo pode ser incluída no documento ou pode ser encontrada em qualquer outro lugar, utilizando uma instrução de processamento para anexar uma folha de estilo ao documento, como no exemplo abaixo, o qual informa que o formato encontrado no arquivo *alunos.ssh* deve ser aplicado ao documento *alunos.xml*.

```
<?STYLESPEC "alunos" "alunos.ssh" ?>
```

Segundo [MOU00], existem atualmente três opções de folha de estilo disponíveis para desenvolvedores XML:

- **CSS** – Cascading Style Sheets, (folhas de estilo em cascata), níveis 1 e 2;
- **DSSSL** – Document Style Semantics and Specification Language, (semântica e linguagem de especificação e semântica de estilos de documentos);
- **XSL** – Extensible Style Language, (linguagem de estilos extensível).

CSS1 e DSSSL são padrões existentes, ao passo que CSS2 e XSL estão em desenvolvimento pelo W3C e grupos correlatos.

As folhas de estilo permitem anexar estilos variados aos documentos, sem prejudicar os benefícios de uma marcação generalizada, de fácil uso e permitindo assim, uma sintaxe para expressar como os documentos XML devem ser apresentados.

2.6.1 CSS – Cascading Style Sheet

Foi desenvolvida originalmente para HTML, e os navegadores que aceitam XML normalmente também aceitam CSS. Uma CSS é um conjunto de regras que diz ao navegador qual fonte, estilo e margem devem ser usados para exibir o texto, como no exemplo abaixo (Figura 20):

```
artigo {
    font-family: Palatino, Garamond, "Times New Roman", serif;
    font-size: 10 pt;
    margin: 5px;
    display: block;
    font-style: italic
        }
```

FIGURA 20. Parte de um exemplo de uma CSS simples, onde aparece uma regra para todo o elemento *artigo do documento*.

Uma folha de estilo CSS é uma lista de regras, onde cada regra começa com um *seletor* e após, as *propriedades* (entre chaves{}), associadas a ele.

Para conectar uma folha de estilo ao documento XML, se usa a instrução de processamento de folha de estilo XML, no entanto, o tipo deve ser adaptado para *text/css*, sendo o próprio navegador o processador do documento, o que pode ser observado na Figura 21., na segunda linha.

```
<?xml version="1.0" encoding="ISSO-8859-1"?>
<?xml-stylesheet href="artigo.css" type="text/css"?>
<artigo pnome="19990101_xsl">
<titulo>Folhas de Estilo XML</titulo>
<data>janeiro de 1999</data>
<secao>
    <p> Envie comentários para.....
</secao>
<secao>
    <titulo>Como funciona a XSL</titulo>
    <p>Folhas de estilo são herdadas da SHML.....
</secao>
</artigo>
```

FIGURA 21. Parte de um documento XML, vinculado a uma folha de estilo CSS.

2.6.2 XSL - eXtensible Style Language

A finalidade da *XSL - eXtensible Style Language* (linguagem de estilo extensível) é tornar-se independente de qualquer tipo de formato resultante, ou seja, uma mesma folha de estilo pode ser usada para XML, para RTF (Rich Format Text), para HTML ou qualquer outro formato.

A XSL é poderosa o suficiente, para lidar com a deleção, reorganização e geração de conteúdo "em paralelo" ao processo de formatação. Instruções sobre a renderização do documento se encontram separadas dos dados propriamente ditos, sendo um dos pontos importantes e favoráveis da XML. Também incorpora a versão padronizada internacionalmente do *JavaScript*, conhecida como *ECMAScript (ECMA-262*), permitindo que através da linguagem de folha de estilo, se possa recorrer ao *JavaScript* para realizar cálculos complexos e de acordo com a necessidade da aplicação.

A folha de estilo da XSL possui o modelo da estrutura desejada como resultado, identificando os dados no documento fonte e os inserindo no template. Diferente do CSS, que decora a árvore XML com informações de formatação, XSL transforma a árvore XML em uma nova árvore, permitindo reordenamento, geração de texto adicional e cálculos, sem modificar o XML original. O documento XML original (fonte), pode ser considerado o documento puro, que poderá ser utilizado por diferentes meios ou pessoas, apenas modificando-se as folhas de estilo.

A especificação XML, foi dividida em dois documentos: *a XSL Transformations Specification (XSLT)*, que trata da sintaxe e semântica da XSL, aplicando folhas de estilo para transformar os documentos, e a segunda parte que se concentra na formatação de objetos XSL, seus atributos e, como eles são combinados, [W3C 99].

Uma folha de estilo XSL é composta por um conjunto de regras conhecidas como *regras de construção,* que especifica como um documento origem XML é convertido em uma hierarquia de formatação de objetos chamados de *objetos de fluxo*. Existem também as *regras de estilo*, que fornecem uma forma abreviada conveniente de se especificar características de formatação, causando efeito *cascata*, ou seja, podem ser aplicadas várias regras de estilo ao mesmo elemento no documento XML de origem.

Uma regra de construção básica em XSL tem a seguinte forma geral:

```
<rule>
        [ patterns  ]
        [  action ]
</rule>
```

A parte *patterns* (padrão) da regra, serve para especificar os elementos no documento XML origem que devem acionar essa regra. A parte da ação específica (*action*) determina o que deve ser feito quando a regra é acionada.

No exemplo da regra abaixo, o resultado será a construção de um objeto de fluxo P, onde serão colocados os resultados das regras de acionamento de todos os elementos secundários no elemento *foo,* após o elemento P é fechado.

O *target-element* (elemento-alvo) é usado para especificar o tipo de elemento, nesse caso foo, que acionará a regra. O P maiúsculo indica um objeto de fluxo, o elemento <children/> em minúsculas é derivado de um conjunto de tags XSL e significa "processar todos os elementos secundários desse elemento agora":

```
<rule>
    <-- Pattern -->
    <target-element type = "foo" />
    <-- Action -->
    <P>
            <children/>
    </P>
</rule>
```

Os padrões e ações são criados usando suas próprias construções sintáticas, como no exemplo abaixo, no qual associa o elemento <title> da DTD a um elemento <P> em um documento XML:

```
<rule>
    <titlle-element type="title"/>
    <P font-size="36pt" font-style="italic">
            <children/>
    </P>
</rule>
```

Essa regra de construção indica que o *target-element title* deve ser apresentado no documento de saída como um parágrafo e que seus filhos (texto e conteúdo) também devem ser processados, conforme definido no element *children.* Os atributos font-size e font-style incluídos no elemento parágrafo criarão um atributo de estilo quando este for apresentado. Esta técnica permite que desenvolvedores, acrescentarem facilmente estilo CSS aos documentos HTML criados a partir de documentos XML

2.6.3 DSSSL - Document Style Semantics and Specification Language

DSSSL, semântica e linguagem de especificação e semântica de estilos de documentos, é um mecanismo de folhas de estilo que funciona com documentos descritos por marcação como uma hierarquia de elementos. As especificações de formatação e de processamento são as de elementos de documentos, e uma especificação visual que é criada para o documento à medida que ele e sua folha de estilos são processados por um analisador sintático XML e por um analisador sintático DSSSL. O DSSSL foi criado para funcionar com os diferentes tipos de mídia nos quais os documentos SGML podem ser propagados, inclusive de forma impressa e vários formatos eletrônicos.

Na realidade o DSSSL pode ser usado para criar diferentes folhas de estilo para o mesmo documento, podendo serem adaptadas para uma ampla gama de tipos de mídia e, dependendo de como o documento for propagado, podem compartilhar estilos comuns, mas tirando proveito de propriedades específicas de mídias em suas apresentações finais.

O DSSSL reconhece que um documento pode ser processado de várias formas, inclusive:

Transformação – para converter um documento XML de uma DTD em outro;

Formatação - rege o tamanho das páginas, margens, tamanho de fontes;

Consulta – usado para identificar seções específicas em um documento XML, semelhantes aos XPointers;

Expressão – usada para criar e manipular objetos.

3 Processamento de Informações em XML

Para ver o resultado da marcação e do conteúdo de documentos e DTDs XML basta processar as informações contidas dentro deles. Um documento XML quando processado, passa por várias etapas antes de o conteúdo ser finalmente exibido. Essas etapas incluem a análise sintática, formatação de folhas de estilo e, finalmente a exibição do conteúdo XML.

Segundo [MOU00], um processador é um módulo de software usado para ler documentos XML e fornecer acesso às estruturas e conteúdos desses documentos. Em XML, este processamento, é feito com analisadores sintáticos e processadores, que pesquisam a estrutura do documento XML e passam os resultados para um aplicativo XML, que por sua vez, faz algo com o conteúdo. O propósito do processamento é interpretar a estrutura, informar a ocorrência de erros e depois passar a interpretação para um outro aplicativo XML.

Existem cinco etapas, para que a estrutura e o conteúdo criado sejam exibidos em um navegador ou enviados para outro aplicativo para processamento adicional, cada uma delas feita por aplicativos específicos para esta finalidade:

1. criar a DTD e o documento XML;
2. passar a DTD e o documento por meio de um analisador sintático de modo que ele possa reunir as informações gramaticais necessárias;
3. processar os resultados da análise sintática e verificar erros;
4. passar os resultados da análise sintática para um aplicativo que consiga estabelecer uma interface com documentos XML;
5. exibir ou enviar as informações de acordo com as instruções de formatação de folhas de estilo para um navegador ou aplicativo XML.

3.1 Editores XML

São usados para criar documentos XML, com recursos específicos para criação de elementos, atributos e entidades e de seus respectivos valores, verificação de erros, além de executar todas as tarefas envolvidas no processo de marcação adequada do documento. Pode-se usar também qualquer editor de textos que salve os arquivos no formato somente texto(*.txt*) ou sem formatação,

3.2 Analisadores Sintáticos XML

Após o arquivo XML ou DTD ter sido criado, é submetido a aplicativos de software e componentes que analisam os documentos quanto à gramática, se são documentos válidos e bem-formados. Esta análise pode ser feita de várias formas, mas normalmente é feita no cliente, o que significa que o servidor praticamente não se envolve neste processo.

O produto ou método a ser utilizado para analisar sintaticamente ou quando deverá ser analisado fica à critério do desenvolvedor, porém alguns analisadores sintáticos fazem parte do aplicativo para exibição final, outros são componentes de software separados.

O analisador sintático compara o código no documento XML com as regras estabelecidas antecipadamente na especificação XML, devendo estar apto a ler caracteres Unicode de 8 e 16 bits e entidades de caracteres. Inicialmente lê o documento XML interno, antes de interpretar DTDs internas e externas, garantindo assim que as declarações locais serão processadas. Quando o analisador sintático está interpretando as informações do documento XML e/ou DTD, são repassadas informações específicas para o próximo aplicativo.

3.3 Parsers

Um parser é a ferramenta mais básica, porém de grande importância, todo o aplicativo XML é baseado em um parser. É o componente de software residente entre o aplicativo e os arquivos XML, com o objetivo de ocultar do desenvolvedor/programador, os detalhes da sintaxe XML.

Existem *parsers* de validação e de não validação, ambos impondo regras sintáticas, porém somente os parsers de validação sabem como validar documentos, comparando-os com suas DTDs.

3.3.1 A arquitetura de um programa XML

A Figura 22 mostra a arquitetura dos programas XML, dividida em duas partes, o parser, que trata o arquivo XML e, a aplicação, que consome o conteúdo do arquivo por meio do parser, [MAR00].

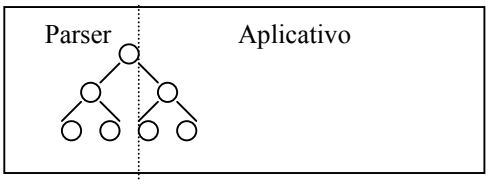

FIGURA 22. Arquitetura de um programa XML.

A aplicação pode ser extremamente simples, como exibir dados na tela, ou mais complexas, como um editor ou um navegador.

O parser e a aplicação devem compartilhar um modelo comum para dados XML, na prática esse modelo, é sempre alguma variação de uma árvore na memória, que corresponde à árvore no documento XML.

O parser lê o documento XML e verifica na árvore de memória. Esta árvore é construída pelo parser conforme os registros da árvore do documento

XML. A aplicação a manipula como se fosse o documento XML e, na realidade para a aplicação, ela é o documento XML.

Existem duas maneiras básicas de se realizar a interface de um parser com uma aplicação , uma utilizando interfaces baseadas em objeto e, outra usando interfaces baseadas em eventos.

Na interface baseada em objeto, o parser cria explicitamente uma árvores de objetos que contém todos os elementos no documento XML. É a interface mais natural para a aplicação, pois recebe uma árvore na memória, que corresponde exatamente ao arquivo no disco, não sendo necessário se preocupar com a sintaxe XML. Se for utilizado parser de validação, a árvore pode ser validada em relação à DTD.

A interface baseada em evento é natural para o parser, mas é complexa para a aplicação, tornando-se a preferência dos programadores devido a sua eficácia. O parser não monta explicitamente uma árvore de objetos, mas lê o arquivo e gera eventos à medida que encontra elementos, atributos ou texto no arquivo. Existem eventos para início de elemento, término de elemento, atributos, conteúdo de texto, entidades, etc.

Na prática as duas interfaces são úteis e atendem a objetivos diferentes. A interface baseada em objetos são ideais para aplicações que manipulam documentos XML, como navegadores, editores e processadores XSL.

As interfaces baseadas em evento servem para aplicações que mantém sua estrutura própria de dados em um formato não XML, como por exemplo, aplicações que importam documentos XML para bancos de dados. Essas aplicações possuem sua própria estrutura de dados, e relacionam uma estrutura XML com sua estrutura interna, sendo assim menos objetos são necessários e menos memória é utilizada.

O autor menciona a necessidade da existência de um padrão entre o parser e a aplicação, permitindo que sejam desenvolvidas aplicações utilizando um parser e após seja distribuído usando outro parser. Os dois métodos diferentes para as interfaces resultam em dois padrões diferentes. O padrão para interface baseada em objetos é o DOM (Document Object Model), publicado e padronizado pelo W3C e o padrão para interfaces baseadas em eventos e o SAX (Simple API).

Os dois padrões não estão em oposição, pois atendem necessidades diferentes, e muitos parser existentes hoje, aceitam as duas interfaces.

3.4 Processadores XML

Após o documento ter sido analisado sintaticamente, um processador lê o documento e oferece a outros aplicativos, acesso aos componentes XML, armazenados dentro do documento XML. Um processador poderia ser um aplicativo de pesquisa usado para recuperar informações contidas em um banco de dados e os resultados obtidos, exibidos em um browser.

Quando um processador intercepta as informações analisadas sintaticamente, os desenvolvedores de software podem acessar rapidamente a estrutura de um documento XML e trabalhar com os componentes de várias

formas. Todos os elementos, atributos, entidades e notações, são então considerados objetos que podem ser manipulados ou exibidos por outros programas através das linguagens de programação e navegadores adicionais.

Como na maioria dos analisadores e processadores XML, o papel do processador (fornecimento de acesso ao conteúdo e estrutura), se sobrepõe ao papel do analisador (checagem da sintaxe), um navegador Web que for capaz de exibir ou interpretar arquivos XML, tem um analisador sintático XML anexado, que analisa os dados e os repassa para o processador, que após identificar os formatos de estilo especificados no arquivo, exibe os dados XML no formato especificados na folha de estilos.

Pode ser incluída uma API-Application Programming Interface-(interfaces de programação de aplicativos) nos processadores para fornecer funcionalidade e serviços de processamento adicionais, e atualmente vem surgindo diversos destes kits de APIs para XML.

Hoje em dia, todos os navegadores implementam completamente todos os recursos presentes na versão 1 da especificação XML, conseguindo entender o que são os arquivos XML e DTDs, como também analisam, processam e exibem seus conteúdos. Isso ocorre porque os documentos XML começam com a instrução de processamento XML, colocada no início do documento para informar ao componente analisador sintático ou processador XML, que essas informações se destinam, na realidade, a serem processadas como um arquivo XML.

Como o XML separa a marcação de apresentação da marcação estrutural, torna-se mais fácil aos desenvolvedores embutir instruções específicas sobre como exibir o mesmo conteúdo XML de diversas formas.

O navegador ou o software de exibição é responsável por efetivamente, descarregar a exibição dos dados em um formato que melhor atenda às necessidades específicas do usuário, permitindo que o usuário veja os mesmos dados de diferentes formas, quando estes forem descarregados no computador do cliente, o que reduz sensivelmente o tráfego no servidor, fazendo parecer como se o navegador estivesse processando as informações mais rapidamente do que com as tradicionais páginas HTML.

Devido a esta flexibilidade na exibição do conteúdo, o desenvolvedor deve especificar modos para os dados serem apresentados, incluindo *folhas de estilo* ao código, que definirá o aspecto de exibição do conteúdo, após terem sidos analisados sintaticamente e processados pelo aplicativo XML.

Após ter sido analisado sintaticamente, o documento XML em si, repassa as informações para um processador, que pode ser uma linguagem de script como o JavaScript ou VBScript, ou até mesmo Java.

Um documento *XML* para que possa ser usado, ou melhor, manipulado, deve ser processado, ou seja o processador deverá entender a estrutura do documento, criar o documento *XML* e fornecer uma API (Application Program Interface). Para a interpretação de documentos *XML*, existem duas *APIs* que podem ser utilizadas, *DOM* (*Document Object Model*) e *SAX* (*Simple API for XML*). *DOM* é um padrão definido pelo W3C que cria uma visão em árvore do documento *XML*.

Segundo [PIM00], existem duas formas de processar documentos XML, de maneira a viabilizar a construção de aplicações que aproveitem o máximo da riqueza de um documento. A primeira consiste da utilização de parsers em conjunto com interfaces padronizadas, que são invocadas através de *callback* à

medida em que se realiza o *parser* no documento, de maneira a realizar as ações previstas na aplicação em função do conteúdo do documento. A outra forma é através do modelo de objetos DOM, que não se limita ao processamento amarrado a ordem em que o parser do documento é realizado.

Processando XML com Componentes Java e ActiveX

Embora o XML separe os dados da apresentação do documentos e a maioria dos navegadores já incorporarem suporte à XML, ainda existem usuários que possuem o navegadores habilitados apenas para HTML. Sendo assim, se faz necessário mecanismos para exibir conteúdos estruturados em XML

Podemos apontar e exibir conteúdo XML, tanto com o ActiveX como com o Java, ambos usados para obtenção de dados XML para navegadores HTML específicos.

Inicialmente, define-se o documento XML, a partir do qual as informações serão analisadas sintaticamente e depois processadas no navegador. Após é criada a folha de estilos, que será referida pelo analisador sintático ActiveX e, a seguir define-se o código que aponta para o componente ActiveX ou Applet Java para o arquivo XML, e define-se um destino no qual as informações serão colocadas, através de um script que especificará exatamente onde as informações deverão ser armazenadas.

Processando XML com JavaScript

Utilizando-se linguagens de script, Javascript por exemplo, é possível passar a estrutura do documento esta linguagem, que pode extrair informações do documento, e exibi-las em uma outra janela em maiores detalhes ou com maiores opções para o usuário.

SAX: Simple API for XML

Atualmente já existem diversos processadores, permitindo ao programador dedicar-se às necessidades específicas de suas aplicações, mas a maioria dos processadores não utiliza um formato padrão para apresentação, aplicação e a publicação do resultado final do seu trabalho.

SAX é uma interface Java simples, que os processadores podem utilizar para esta tarefa, sendo, então, uma classe que implementa a interface *org.xml.sax.Parser*.

O parser, ao percorrer o documento, realiza chamadas para os métodos implementados na aplicação. A interface SAX pode ser utilizada por aplicações que realizam tarefas relativamente simples ou que necessitem processar os documentos de modo seqüencial, devendo estar disponível todo o documento, o que é conseguido através do DOM. O uso do DOM e do SAX em conjunto é justificado pelo fato de que o modelo DOM não define algumas interfaces especificadas por SAX.

Segundo [ABI00], é uma padrão para análise sintática de dados XML. Um analisador sintático de SAX lê o fluxo de alguns dados em XML, analisa-os e

detecta eventos quando interpreta as marcas. Estes eventos são enviados de volta para a aplicação , proporcionando assim, uma simples API para manipular XML.

DOM – Document Object Model

No contexto do DOM (Modelo de objetos do documento), os documentos XML são representados em uma estrutura de árvore que reflete a sua estrutura lógica hierárquica. É uma API que define uma forma de acesso e manipulação à árvore de objetos correspondente ao documento, permitindo que programas ou scripts acessem e manipulem dinamicamente a estrutura, conteúdo e estilo de apresentação de documentos XML e HTML.

Foi proposto para facilitar a computação sobre documentos: recebe um arquivo XML e disponibiliza sua representação como uma árvore de objetos para aplicações. Suporta a manipulação da estrutura e conteúdo do documento, permitindo que programas possam adicionar, modificar ou apagar nós da árvore de objetos a ele associada, permitindo que programas possam utilizar os documentos de modo mais abrangente do que o suportado por SAX ou XSL.

É concebido como uma interface independente de plataforma, padrão para que possa ser utilizada por uma grande variedade de ambientes e aplicações, portanto, não especifica como seus métodos devem ser implementados, limitando-se apenas à descrição da interface.

Além da definição em IDL – Interface Definition Language, inclui a especificação de interfaces (APIs), para utilização em programas Java e EMACScript, ou seja, os métodos que podem ser utilizados por esses programas de modo a acessar os objetos de um documento. Diz-se que o DOM acrescenta ao padrão destas linguagens toda a funcionalidade que um programa precisa para acessar e manipular documentos, como inserção, modificação, exclusão e a navegação pelo documento.

DOM é uma maneira de descrever um documento XML para um outro aplicativo ou linguagem de programação, em uma tentativa de manipular ainda mais as informações da maneira desejada. Pode ser usado para inserir fontes de dados XML com navegadores, usando-o para coletar informações sobre a estrutura do documento e depois, para criar conexões entre a estrutura do documento e outros aplicativos ou fontes de dados.

O documento XML, que representa todo o documento-fonte XML, pode ser acessado por meio de um script, colocado em uma página Web, para carregar o documento, pois este é visto apenas como um objeto. Para isso pode-se criar um objeto *ActiveX*, e usando o método *load* para carregar o documento.

Como XML tem um padrão de estruturação para formação de seus documentos, o DOM pode acessar qualquer documento XML, e o documento representa um objeto que pode ser manipulado, significando que se pode usar e reutilizar os scripts para acessar documentos XML, pois existe pequenas modificações entre os documentos.

O DOM usa três objetos para acessar o arquivo XML, o documento XML, o nó XML e a lista de nós XML.

O nó em XML representa um nó dentro de um documento XML, como por exemplo um elemento, qualquer instrução de processamento, a declaração XML, atributos e etc.

A lista de nós representa um conjunto de nós, existindo várias propriedades e métodos usados na lista de nós XML, como por exemplo:

- propriedades: *legth*
- métodos: *CurrentNode, Item, MoveTo, MoveToNode, NextNode, previousNode;*

Ao usarmos o DOM podemos atribuir vários objetos XML a variáveis, que podem ser acessadas e manipuladas por uma série de aplicativos e, assim que tivermos criado um objeto documento XML, podemos acessar informações sobre este objeto, usando as propriedades e métodos do DOM.

XML também consegue conectar dados em documentos XML a aplicativos como bancos de dados para tornar as informações mais acessíveis.

4 Aplicativos XML

Com o surgimento do XML, é possível criar aplicativos inteiramente novos, não por meio de métodos de programação convencional, mas sim por meio da definição da estrutura contextual do documento.

Atualmente, existe um número bem razoável de aplicativos XML e, muitos dos navegadores atuais, já incorporam algumas funções XML, com é o caso do Internet Explorer, que já incorpora o CDF.

A W3C afirma o baixo nível de complexidade da sintaxe para representar dados estruturados, permitindo a utilização de uma mesma sintaxe para um largo número de aplicações, o que pode ser verificado na Figura 23, em que XML é apoiada ou escorada por um grande número de linguagens de marcação e aplicações.

Inicialmente suporta sem nenhum problema, HTML, expressões matemáticas que podem ser inseridas utilizando-se *MathML*. O *RDF – Resource Description Format*, permite a representação das propriedades da Web, como imagens, documentos e a relação entre eles. O *PICS – Plataform for Internet Content Selection*, remodela o RDF e traz o significado advindo dos rótulos para o documento, indicado para crianças. E a *SMIL – Synchronized Multimedia Integration Language*, que consiste em uma aplicação declarativa da linguagem programada para apresentações multimídias na Web.

Esse amplo suporte, faz com que a XML tenha um crescimento rápido, surgindo aplicações nas mais diferentes áreas, usada para armazenar ou processar dados. Essa flexibilidade é ideal para o intercâmbio de dados para um processamento futuro em outra máquina, dispondo de classes e métodos que são criados através de linguagens como *Java, ECMAScript, VBScript e* C^{++}.

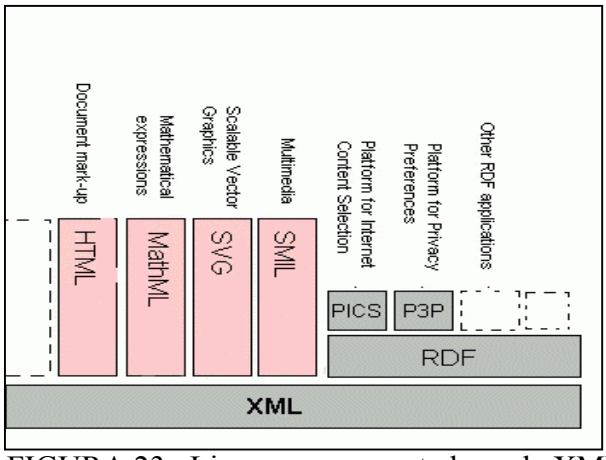

FIGURA 23. Linguagens suportadas pela XML.

Segundo [GRA 99], a XML embora uma linguagem recente, possui um conjunto bastante diversificado de aplicações, fato que é favorecido pela XML ser uma subdivisão da SGML, que existe desde 1986, tornando-se uma tecnologia viável para geração de aplicações na Web.

Os aplicativos XML seguem a sintaxe XML e, para usá-los basta familiarizar-se com seus elementos específicos e seus atributos.

Segundo [MOU00], estes aplicativos se encaixam em sete categorias de aplicativos, mencionadas a seguir.

4.1 Aplicativos para a Web e para a Internet

São os que trabalham via Internet ou Web para enviar informações de conteúdo e de programação, ou para melhorar a comunicação entre computadores através da Internet. Os aplicativos abaixo são considerados aplicativos voltados para a Web:

- *Channel Definition Format* (*CDF*) – desenvolvido pela Microsoft, é um aplicativo XML que possibilita aos desenvolvedores usar uma série de mecanismos de remessa para publicar conjuntos de informação, chamados canais, de qualquer servidor Web para qualquer aplicativo compatível com a Internet. Acompanha o navegador Internet Explorer, e também pode ser acessado por meio do Netscape Communicator.
- *Open Software Description* (*OSD*) – também desenvolvido pela Microsoft, usado para descrever componentes de software, versões de software e a estrutura implícita de pacotes de software e seus componentes para remessa através de uma rede. Pode operar em conjunto com o CDF, para atualizar softwares através da Internet ou de uma Intranet.
- *Distributed Authoring and Versioning on the World Wide Web* (*WebDAV*) – é uma especificação de aplicativo, ainda em fase preliminar de desenvolvimento, que se destina a definir semânticas e métodos HTTP, para criar, eliminar, consultar e editar páginas Web remotamente.
- *HTTP Distribution and Replication Protocol* (*DRP*) – aplicativo que visa aumentar a eficiência e a confiabilidade do emprego do protocolo HTTP para distribuição de dados através da Internet.
- *Wireless Aplication Protcol* – protocolo que oferece padrões para transmissão via redes sem fio e escalonamento por meio de várias opções de transporte e tipos de dispositivos.
- *UML Exchange Format* – foi criado por desenvolvedores de software, como um mecanismo para transferência de modelos UML (Unified Modeling Language), permitindo que os desenvolvedores espressem, publiquem e façam o intercâmbio de modelos UML de forma universal.
- *Web Interface Definition Language (WIDL)* – aplicativo orientado a objetos, que usa protocolo WIDL para automatizar muitas funções na Web, inclusive a entrada de dados, distribuição e intercâmbio de documentos entre vários aplicativos.

4.2 Aplicativos de Metadados e de Arquivamento

Trabalham com determinados tipos de conjuntos de dados, informações sobre estes dados e intercâmbio deles com outros sistemas., entre alguns podemos citar:

- *Resource Description Framework* (*RDF*) – especificação de estrutura que suporta metadados e atividades baseadas na Web como mapas de sites, avaliações de conteúdo, conjuntos de dados de mecanismos de busca e autoria distribuída.
- *Meta Content Framework* (*MCF*) – framework desenvolvido inicialmente pela Apple e depois aperfeiçoado pela Netscape, usa XML para descrever estruturas de organização da informação.
- *Web Interface Description Language* (*WIDL*) – é uma sintaxe de metadados que define APIs para dados e serviços da Web, baseado em XML. Manipula outros tipos de dados, permitindo assim, aos desenvolvedores conectar dados XML a outros tipos de dados ou aplicativos baseados na Internet. , usado com os WebMethods como um método para automação na Web.
- *IMS Metadata Specification* – usa o XML para oferecer remessa efetiva de materiais de treinamento de alta qualidade através da Internet. Também suporta o gerenciamento de materiais e tipos de dados relativos a sites Web. Foi concebida basicamente para atender estudantes on-line, criar relatórios do progresso obtido, para o intercâmbio de registros através da Internet e para operar com vários sistemas de aprendizado administrativos.
- *Encoded Archival Description* (*EAD*) – é uma DTD completa usada para desenvolver um padrão de codificação não proprietário para documentos de biblioteca, de modo que qualquer tipo de informação solicitada possa ser encontrada em bibliotecas e museus.
- *Genealogical Data in XML* (*GedML*) – A DTD GedML foi criada para fornecer um método padrão para apresentação, intercâmbio e manipulação de dados genealógicos com outros usuários através de uma rede, com objetivo de tornar a pesquisa de dados genealógicos mais fácil.
- *XML-Data* – destina-se a efetuar o intercâmbio de dados estruturados e em rede na Web, como por exemplo descrever assinaturas digitais.

4.3 Aplicativos Multimídia

Utilizam a linguagem e a sintaxe XML para criar modos padronizados de apresentação de informações através da Web, particularmente imagens, vídeo e voz digitalizada, entre alguns podemos citar:

- *Synchronized Multimedia Integration Language* (*SMIL*) – especificação recentemente aprovada pelo W3C, destina a ajudar a remeter o conteúdo multimídia para a Web. Permite aos desenvolvedores criar e liberar conteúdo do tipo TV com requisitos de largura de banda baixa. Acrescenta também recursos de links ao conteúdo multimídia tornando possível aos usuários se movimentarem rapidamente para outros conteúdos da Web, e sincronizar as apresentações.
- *Precision Graphics Markup Language* (*PGML*) – é uma linguagem gráfica 2D, oferecendo especificações de elementos gráficos de precisão e vetoriais. Usando modelamento de imagens comum, tanto para o formato PostScript como para o formato PDF.
- *Java Speech Markup Language (JSML)* – permite aos desenvolvedores a criação de aplicativos que anotem texto para reprodução por meio de sintetizadores de voz, por meio de *API Java Speech*. Este formato fornece informações detalhadas sobre como o texto deve ser reproduzido em termos de fala por meio do sintetizador.

4.4 Aplicativos voltados às Áreas de Negócios e Finanças/Comércio

Permitem aos consumidores e empresas, se comunicarem utilizando uma linguagem padronizada, para seus aplicativos financeiro, tornando-se assim independente da instituição financeira em questão. No momento, existem algumas linguagens em desenvolvimento, como:

- *Open Financial Exchange* (*OFX*) – formato de dados usado para representar como informações financeiras podem ser intercambiadas entre um servidor de serviços financeiros e o software do cliente, como por exemplo, o navegador.
- *XML/Eletronic Data Interchange* (*XML/EDI*) - é o *Intercâmbio Eletrônico de Dados* (*EDI*), entre duas ou mais fontes de dados da Web, no qual os documentos XML são considerados banco de dados e as DTDs, o esquema de banco de dados. Uma aplicação EDI, deve realizar as seguintes tarefas: - extração dos dados de documentos XML; - conversão de dados entre banco de dados relacionais e orientados a objeto; - transformação de dados de uma DTD para outra DTD diferente; - integração de múltiplas origens de dados XML. Proporciona uma maneira de integrar dados de diferentes tipos, como por exemplo, uma fatura, uma solicitação de um exame de saúde, uma transação de um projeto, uma transação utilizando uma *API*, automação da Web, um portal, um

documento de *Workflow* ou até mesmo uma mensagem que pode ser, procurada, decodificada, manipulada e visualizada de maneira consistente e correta com a utilização do dicionário da EDI e, ter o vocabulário estendido via on-line..

- *Open Trading Protocol* (*OTP*) – desenvolvido para fazer com que o comércio em larga escala de papéis negociáveis e de ações via Internet se tornasse ao mesmo tempo conveniente e seguro.
- *Information & Content Exchange* (*ICE*) – se baseia no XML para fornecer a troca e o gerenciamento automático e controlado de arquivos on-line entre parceiros comerciais, dando às empresas a possibilidade de transferência e compartilhamento de informações.

4.5 Aplicativos Científicos

Permitem aos cientistas e matemáticos trocar dados de uma maneira padronizada, usando uma linguagem que permite representar por exemplo os elementos construtivos de um átomo, ou a exibição de equações matemáticas de forma apropriada, citamos a seguir, alguns:

- *Mathematical Markup Language* (*MathML*) - já é uma recomendação do W3C, fornecendo uma maneira padrão para descrição de equações e símbolos matemáticos em páginas Web.
- *Chemical Markup Language* (*CML*) – fornece uma maneira de descrever informações moleculares e gerenciar uma ampla gama de problemas e equações químicas dentro de uma única linguagem. Ajuda os químicos a manipularem átomos e moléculas como elementos.
- *Bioinformatic Sequence Markup Language* (*BSML*) – ainda em fase de análise, visando fornecer um método padrão para codificação e exibição(via Internet) de informações de seqüências de proteínas, DNA e RNA entre programas e dados.
- *Telecomunications Interchange Markup* (*TIM*) – usada para fornecer um mecanismo padrão às indústrias destinadas a provisão, obtenção e emprego de equipamentos, produtos e serviços de telecomunicação.

4.6 Aplicativos Educacionais

São aplicativos destinados a criação de aplicativos para o aprendizado na Web, entre alguns pode-se citar o *TML - Tutorial Markup Language* (linguagem de marcação tutorial), que foi o primeiro a surgir desta categoria, que separa o conteúdo semântico de uma questão de sua formatação ou layout de tela, permitindo assim, que as perguntas e respostas possam ser pesquisadas, catalogadas e calculadas facilmente. Permite ao desenvolvedor determinar itens como quantas tentativas um candidato pode fazer para responder a uma pergunta corretamente, quais as perguntas, quais as opções serão incluídas nas repostas, dicas para as perguntas.

4.7 Aplicativos voltados a linguagens

Se destinam à linguagem de computadores e ao intercâmbio entre sistemas computacionais. São os aplicativos mais encontrados em termos de aplicativos baseados em XML, ajudam as máquinas a tomar decisões sobre os dados armazenados em páginas Web e em bancos de dados, alguns:

- *Translation Memory eXchange* (*TMX*) – tem o propósito de permitir o intercâmbio mais fácil de dados de memórias de tradução entre ferramentas e/ou fornecedores de produtos de tradução sem nenhuma ou pouca perda de dados críticos durante o processo.
- *Ontology Markup Language (OML)* – permite aos autores das páginas Web, anotar as suas páginas, de forma que possam ser lidas por máquinas e processadas com software de agentes inteligentes.
- *Conceptual Knowledge Markup Language (CKML)* – segue a filosofia do *CKP – Conceptual Knowledge Processing* (processamento de conhecimento conceitual), usada para representar conhecimento e análise de dados, que servem para criar modelos de pensamento relacional, julgamento e tomada de decisão. Trata-se de uma extensão do aplicativo *OML.*
- *OpenTag* - ajuda a criar uma maneira padronizada de codificar diversos tipos de arquivos por meio do emprego de um método de marcação comum.

Segundo [PFE 98], XML descreve os dados e preserva seu significado, em todas as etapas do processamento, permite o seu uso para diferentes aplicações, devido a sua flexibilidade de formato, sendo usada para descrição do formato dos dados, intercâmbio do formato dos dados, armazenamento do formato dos documentos e criação de *triggers* para processamentos especializados.

A indústria atualmente está empenhada na criação de DTDs públicas para propósitos particulares bem definidos, e as pesquisas em XML estão baseadas em uso das aplicações em HTML já existentes, *middle-tear processing*, documentos estruturados e inter-operabilidade entre os diferentes formatos de documentos existentes.

O autor também menciona que significativas pesquisas podem ser aplicadas em dados XML, e os resultados podem ser interpretados de maneira diferente, dependendo do dispositivo de saída, observando-se que o processador pode estar no cliente, no servidor ou em ambos.

As *tags* definem o que os dados significam, utilizando-se um vocabulário natural, o que faz com que a XML venha se tornando uma linguagem padrão e tão próspera para a Internet. e que na perspectiva atual, quase todos os tipos de dados podem ser representados em XML, com uma gramática especial para descrever a estrutura, porém algumas áreas tem benefícios imediatos com o uso da XML, entre elas:

a) *e-commerce*: nesta área, os documentos XML, descrevem ordens, transações, inventários, faturas, e são acessados por bancos, empresas fabricantes, consumidores, entre outros;
b) pesquisas independente de plataforma: se os dados estiverem codificados em XML, e descrevendo o seu significado atual, customizações podem ser realizadas sem levar em consideração a plataforma dos servidores e dos clientes;
c) acesso aos dados: podem ser imediatamente acessados ou incorporados em novas aplicações;
d) aplicações simplificadas: é necessário que os aplicativos leiam diferentes formatos de documentos, com o uso de XML esse processo é simplificado, aumentado o compartilhamento dos dados, aumentado ou acelerando as conexões entre comércio e consumidores;
e) como nos objetos *Java* ou *Javabeans*, XML é *auto-descritível* ou *auto-declarável*;
f) possui uma *API*, própria para acessar a estrutura lógica e o conteúdo do documento XML;
g) os *parsers* XML, possuem o *API DOM*, trazendo benefícios para os programadores como a redução do código do programa, e permitem que este *API* use XML para gerar ou escrever XML.

5 Linguagens de Consulta para XML

A linguagem XML vem se firmando como padrão para o intercâmbio de informação e produção de documentos eletrônicos, dessa forma as linguagens de consulta utilizadas para estes documentos XML, passaram a receber atenção especial por parte do pesquisadores.

Consultar documentos XML representa um desafio para a comunidade de banco de dados, [KAD99], dada a perspectiva de que grandes volumes de dados estejam disponíveis nesse formato, gerando a necessidade de ferramentas de consulta que sejam , ao mesmo tempo, flexíveis o suficiente par compreender a heterogeneidade dos documentos e poderosas a ponto de extrair informações com rapidez e correção.

Segundo [MAI98] e [BAR98], existem várias linguagens para consultar documentos XML, as quais podem ser comparadas utilizando-se diferentes critérios, sendo que os mais importantes são aqueles que dizem respeito aos bancos de dados.

A seguir citamos algumas das características acima mencionadas, que são desejáveis para as linguagens:

- **resultados em XML** – espera-se que o resultado de uma consulta de uma linguagem para XML seja também, em XML, permitindo que visões possam ser definidas por meio de uma única consulta. Assim torna-se transparente para as aplicações se estão trabalhando com resultados de consultas anteriores ou com os dados originais, proporcionando maior flexibilidade às consultas;
- **permitir um mínimo de operações** – devem oferecer no mínimo operações de *seleção*, que permite incluir no resultado somente os elementos necessários, de *extração*, para extrair elementos específicos no documento, de *redução*, para remover sub-elementos de um elemento, de *reestruturação*, para construir um novo conjunto de instâncias de elementos para armazenar os dados resultantes da consulta e de *combinação*, que permite combinar dois ou mais elementos em um só. Além destas, devem também expressar *joins*;
- **dependência de DTD** – A linguagem deve ser utilizável mesmo na ausência de uma DTD, uma vez que sua presença não é obrigatória, porém se existir e estiver disponível, a linguagem deve ser capaz de utilizá-la, a fim de identificar se a consulta foi bem formulada, além de preparar uma DTD para o resultado da consulta;
- **ordem dos elementos** – é importante uma vez que um documento só é considerado bem formado se os elementos estiverem devidamente encadeados, visualizados como uma árvore sempre que deverá sempre ser mantida em ordem pela linguagem, devido a importância da ordem dos elementos;
- **conhecimento de namespaces, XLink e Xpointer** – fornece às linguagens a capacidade de integrar de forma eficiente diferentes fontes de dados XML, bem como consultar bases de documentos conectados por hyperlinks;

- **satisfatória para metadados** – deve ser capaz de consultar de forma eficiente os metadados, que são tão importantes quantos os dados propriamente ditos, das linguagens existentes para XML nenhuma deixa explícita a possibilidade de consultar DTDs ou não. Entretanto existem outros formatos para metadados como RDF e o XML Schema;
- **existência de uma álgebra** – uma linguagem de consulta XMl deve possuir uma álgebra equivalente, para que se possam utilizar técnicas de otimização como aquelas desenvolvidas para bancos de dados relacionais.

A seguir são citadas as principais linguagens de consulta para XML encontradas:

5.1 XML-GL

É uma linguagem proposta por [CER 98], baseada em um modelo de dados chamado XML-GDM (XML Graphical Data Model), utilizado para representar as DTDs e os próprios documentos XML O modelo possui uma representação gráfica, tanto para o documento como para as DTDs, permitindo a criação de consultas em XML-GL com uma interface *Wysiwyg*.

Na Figura 24, podemos observar o modelo XML-GDM para a DTD da Figura 16, na qual podemos observar a distinta representação que é dada aos componentes da DTD. Os retângulos indicam elementos não terminais, que se conectam por setas a pequenos círculos vazados, que representam elementos terminais, ou a outros elementos não terminais. Os atributos são representados por setas que se originam do elemento ao qual eles pertencem e apontam para pequenos círculos hachurados. Elementos alternativos são ligados por um arco, rotulados pela palavra *xor*. Atributos do tipo IDREF são conectados a um tipo especial de elemento, pré-definido pelo modelo, chamado ANY.

A cardinalidade dos elementos é representado por números mínimos e máximos colocados ao lado das setas que conectam os elementos e atributos. Assim, uma marcação (0:1) indica que um elemento é opcional, podendo ocorrer zero ou, no máximo, uma vez.

A ordem dos elementos é indicada da seguinte forma: o primeiro elemento tem a seta cortada por uma pequena linha, e os demais são desenhados na direção contrária ao movimento dos ponteiros do relógio.

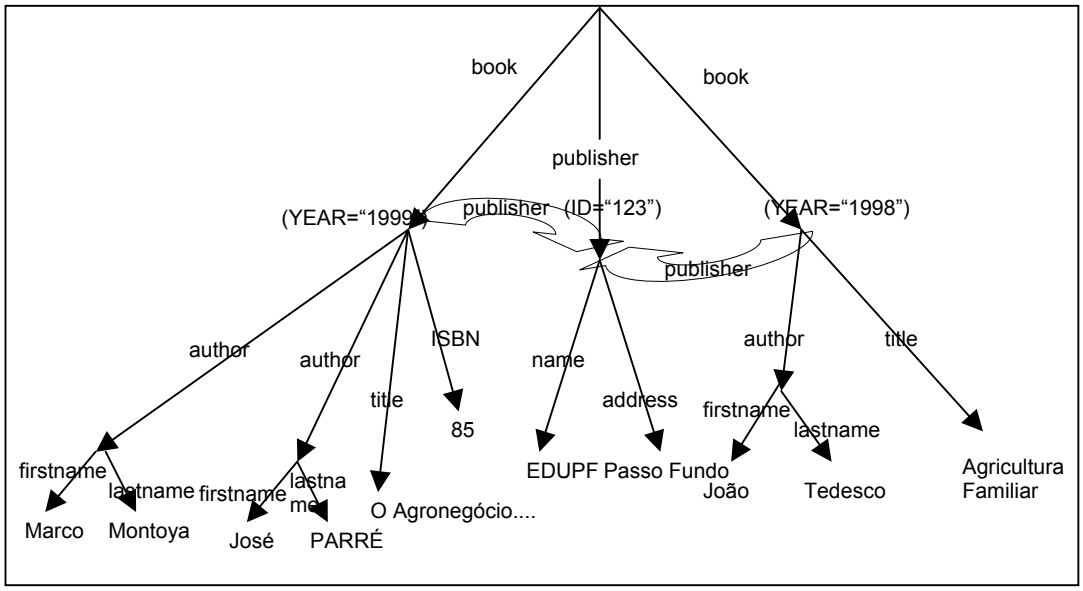

FIGURA 24. Modelo XML-GDM para a DTD da Figura16.

Fonte: [KAD99].

Segundo [CER 98], uma consulta XML-GL pode ser aplicada a um documento ou a um conjunto de documentos, produzindo sempre como resultado um documento XML, composta de quatro partes:

- a *extract part* identifica o escopo da consulta, indicando tanto os elementos de origem como os elementos que o compõem, (corresponde a cláusula *from* do SQL);
- *a match part* é opcional, e especifica condições lógicas que os documentos de origem devem satisfazer para se tornarem parte do resultado da consulta, (*where* do SQL);
- *a clip part* especifica quais são os documentos que constarão do resultado da consulta, (*select* do SQL);
- a *construct part é* opcional, e especifica quais subelementos, dos elementos extraídos pelo *match*, devem ser mantidos no documento final, semelhante ao *create view* do SQL, que permite a criação de novos elementos, definição de novos links, e a reestruturação de informações locais a um dado elemento;

Deve-se ressaltar que XML-GL não é apenas uma interface gráfica, mas sim uma linguagem de consulta completa, na qual as consultas são representadas de forma gráfica. A representação inclui um par de grafos XML-GDM, dispostos lado a lado e separados por uma linha vertical. O grafo do lado esquerdo representa as partes *extract* e *match*, enquanto que o grafo do lado direito, corresponde as partes *clip* e *construct*, representando a DTD do resultado, que está sempre presente mesmo que a consulta seja aplicada em documentos *well-formad*, sem uma DTD.

A Figura 25 mostra o tipo mais simples de consulta XML-GL, chamada *extract-clip*, no lado esquerdo, a parte *extract*, a origem é o endereço mencionado e significa que desse diretório são selecionados todos os elementos do tipo *book* dos arquivos *.xml* que começam com "*bib*". Já no lado direito, o

retângulo superior, rotulado como *book*, indica o elemento inicial, enquanto que o inferior, sem rótulo, indica qualquer elemento. O asterisco indica que todos os elementos, independente do nível de aninhamento, serão selecionados.

A expansão do tipo básico de consulta permite que sejam estabelecidos quaisquer tipos de predicados de seleção sobre os elementos dos documentos de origem, assim como também pode ser produzida qualquer combinação de elementos no resultado, permitindo também *joins* e *agregações*.

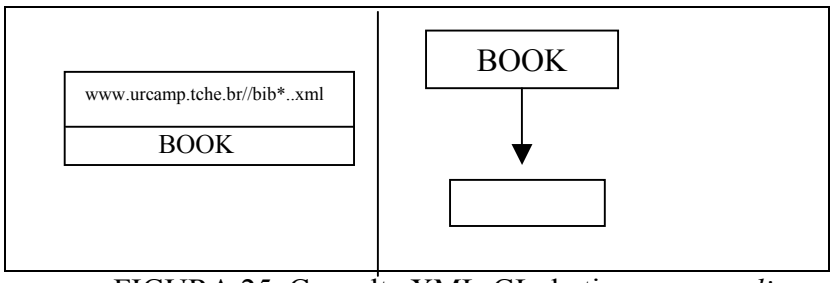

FIGURA 25. Consulta XML-GL do tipo *extract-clip*.
Fonte: [KAD99].

5.2 LOREL e OQL

Segundo [ABI 99], a linguagem LOREL utiliza como base a OQL (Object Query Language), com modificações e extensões para suportar dados semi-estrutrados, sendo que inicialmente, utilizava como base um modelo de dados chamado OEM – Object Exchange Model, que serve como base para várias outras linguagens, mas atualmente a linguagem trabalha com XML.

O autor menciona duas novidades usadas na Lorel: uso extensivo da *coercion* (coerção), para auxiliar o usuário precisamente na modelagem da OQL, vez que é inadequada para dados semi-estruturados, e possui um poderosa maneira de representar expressões de caminho (path expressions), a qual permite uma forma flexível de acesso navegacional declarativo, e são particularmente apropriados quando os detalhes da estrutura não são conhecidos pelos usuários. Lorel também inclui uma atualização declarativa da linguagem.

Foi implementada como uma linguagem de consulta pelo *Sistema Gerenciador de Banco de Dados Lore* em Stanford, a qual permite armazenamento, consulta e atualizações em dados semi-estruturados.

No contexto do modelo de dados da Lorel, um documento XML, é um grafo onde cada elemento XML, é um par *<eid,valor>*, onde eid é um identificador único de elemento, e o valor pode ser do tipo *atômico* ou *complexo*, contendo os outros componentes a seguir:

a) um *valor de string* tag, correspondendo ao tag XML para o elemento;
b) uma lista ordenada de pares *nome-de-atributo/valor atômico*, onde cada nome de atributo é uma string cada valor atômico possui um tipo, como *integer, real, string,* ID, IDREF ou IDREF;
c) uma lista ordenada de *subelementos cruzados*, na forma *<rótulo,eid>*, onde rótulo é uma string e os subelementos são inseridos por meio de atributos do tipo IDREF ou IDREFS;

d) uma lista de *subelementos normais* na forma <rótulo, eid>, onde rótulo é uma string e os subelementos normais são introduzidos por meio do *aninhamento léxico* do documento XML..

Os elementos básicos das consultas em Lorel são as expressões de caminho (*path expressions*), como por exemplo, *biblio.book.author.lastname*. Além da forma mais simples, as expressões de caminho também podem incluir *wildcards* e expressões regulares, bem como *qualificadores de expressões de caminho*, para especificar se devem ser pesquisados somente subelementos ou atributos, utilizando os símbolos ">" e "@", respectivamente.

Entre as principais características da Lorel está a possibilidade da coerção automática de tipos e a comparação de conjuntos de elementos com um elemento único, o que diminuem os números de erro resultantes da conversão de tipos.

Para permitir todos os tipos de comparações possíveis, a linguagem fornece uma série de funções pré-definidas além de uma semântica que é utilizada quando não são fornecidas as funções. Podem ser feitas consultas que concatenem os conteúdos dos elementos, eliminando as *tags* e comparações de identificadores de elementos. Na consulta abaixo, o conteúdo dos elementos nome e sobrenome do autor são concatenados, a fim de compará-los com um texto.

```
select biblio.book.tittle
where biblio.book.concatene(author)="Marco Montoya"
```

Outro recurso que a linguagem oferece são os *range qualifiers* (qualificadores de extensão), que permitem selecionar quais os elementos que serão recuperados, por seu número de ordem em relação ao elemento selecionado. As duas consultas abaixo são exemplos do uso de *range qualifiers*, onde na primeira consulta retorna os dois primeiros autores de cada livro selecionado e, a segunda consulta retorna os dois primeiros autores de todos os livros selecionados.

```
select A from biblio.book B, B.author[1-2]
select A[1-2] form biblio.book B, B.author
```

Com relação à ordenação, além da ordem natural dos identificadores, também pode ser especificada na consulta pela cláusula *order by*, a *expressão order-by-document-order*, retorna os elementos ordenados conforme estavam no documento original.

A linguagem ainda oferece dois recursos para facilitar a reestruturação dos documentos, a cláusula *with*, que combina elementos especificados com os da cláusula *select*, permitindo montar novos conjuntos de dados; e as funções S*kolem*, que recebem como argumento um conjunto de elementos e/ou atributos, gerando um único elemento para cada combinação desses elementos ou atributos.

Também suporta atualizações em documentos XML, através de uma linguagem de atualização, permitindo a criação de novos documentos e atributos além das restruturações normais nos documentos.

LOREL é a única linguagem com extensão da OQL e que manipula dados semi-estruturados.

Os exemplos de consultas abaixo demonstram a simplicidade da linguagem Lorel sobre dados semi-estruturados, onde no primeiro exemplo é procurado o endereço de todos os restaurantes que possuem o CEP "92310", e no segundo, são obtidos o nome e o CEP de todos os restaurantes mais baratos. O ponto de interrogação ("?") após .address é opcional, já o "%" combina qualquer subobjeto de *restaurant* e compara com o operador *grep* que retornará verdadeiro se encontrar a string "cheap" em qualquer sub-objeto:

```
select Guide.restaurant.address
where guide.restaurant.address.zipcode=92310


select Guide.restaurant.name,
        Guide.restaurant(.address)?.zipcode
where Guide.restaurant.% grep "cheap"
```

Segundo [VIS 98], a OQL é uma linguagem declarativa, o oposto das demais linguagens procedurais, a qual segue alguns princípios de operação da ODBMS como por exemplo:

- transformações algébricas, expressões fatoriais, prioridades de seleção e expressões constantes;
- o caminho de acesso é indexado ou clusterizado.

Os operadores em OQL podem ser:

- numerais (+ - * / mod abs)
- comparativos (= != > <)
- booleanos (not and or)
- construtores
- objetos: Employee(name: "John, slary:15000)
- coleções: Vector(1, 2, 3)
- estruturas: strcut(name: p.name, age:p.age)
- coleções literais: set(1, 2, 3)

Os operandos da linguagem OQL podem ser do tipo *constantes* (1.0; true; "A string"; nil), *named objectos* (People; president) e *variáveis* (p; p.age ou p-> age; I(i), com a mesma sintaxe para atributos e métodos.

A seguir um exemplo de uma consulta em OQLsimples, após com a inclusão da linguagem C++ e, por último, com agregação.

```
select struct(n: p.name, s: p.Sex)
from People as p
where age < 18

d_oql_execute(teenagers,
    select p from People as p where p.age < 18"

select *
form Employess as e
group by e.departament as departments
order by avg(select e.salary from partition)
```

A relação entre *Lorel/OEM* e *OQL/ODMG* pode ser observada na Figura 26, onde Lorel pode ser traduzida sintaticamente como uma extensão da OQL, o que inclui objetos heterogêneos, variáveis de busca e *wildcards*.

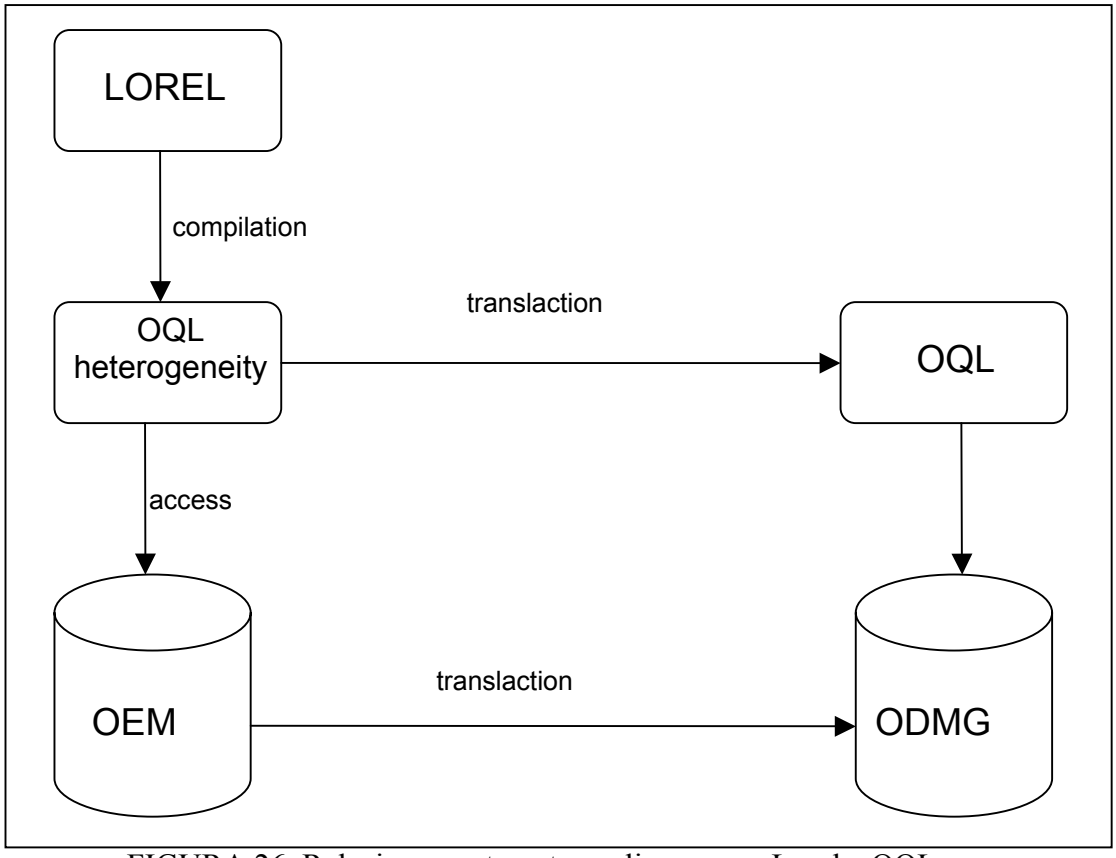

FIGURA 26. Relacionamento entre as linguagens Lorel e OQL.
Fonte: [VIS98].

5.3 XQL

Segundo [ROB 99], a XML Query Language (XQL), foi desenvolvida pelo *Texcel Research, webMethods e Microsoft*, com o propósito de realizar consultas em documentos XML.

A tabela 3.1 mostra as diferenças entre SQL e a linguagem XQL e a Figura 27, mostra um exemplo de uma entrada de uma consulta simples em XQL (*search context*), a consulta propriamente dita (query) e, em seguida o resultado desta consulta (*result*), observando que ambos, o *search context* e o *result set*, neste exemplo, contém apenas um nodo, por isso são iguais.

Tabela 3. Comparação entre SQL e XQL.
Fonte: [ROB 99].

SQL	XQL
O banco de dados é uma coleção de tabelas.	O banco de dados é um ou mais documentos XML.
As consultas são feitas em SQL, usando as tabelas como modelo básico.	As consultas são feitas em XQL, usando a estrutura da XML como modelo básico.
A cláusula FROM determina as tabelas que deverão serem examinadas pela consulta.	A consulta é obtida através de um conjunto de nodos de entrada de um ou mais documentos, e são examinados os nodos e seus descendentes.
O resultado da consulta é uma tabela contendo um conjunto de linhas.	O resultado da consulta é um conjunto de nodos de documentos XML, os quais poderão ser envolvidos em um nodo raiz para gerar um documento XML vem formado (well-formed).

```
Search Context:
<novel>
    <front>
        <title> The Heart of Darkness </title>
        <author> Joseph Conrad </author>
    <front>
</novel>
Query:
novel
Result:
<novel>
    <front>
        <title> The Heart of Darkness </title>
        <author> Joseph Conrad </author>
    <front>
</novel>
```

FIGURA 27. Exemplo de uma consulta simples em XQL.
Fonte: [ROB99].

Uma importante motivação para a XQL, é que a utilização de XML, implica no modelo de dados (*data model*), o qual não tem nenhuma semelhança com um banco de dados relacional. O primeiro passo na construção de uma consulta é entender a sua estrutura implícita. Cada nodo de um documento XML tem um tipo, ou contém um valor, e são acrescidos dos nomes dos atributos e elementos , que por sua vez, devem ter prefixos e *namespaces (espaços de nomes)* associados.

As consultas em XQL sempre atuam sobre um determinado *contexto,* o que de uma forma sintética, pode ser considerado o escopo da consulta dentro da árvore do documento XML. Assim, por exemplo, o resultado de consultas em que se procura o elemento *author* são diferentes quando o contexto é um nodo *book*, ou quando o contexto é a raiz da árvore. Neste último a consulta deveria ser realizada sobre todo o documento XML, enquanto que no anterior não. As consultas em XQL são realizadas utilizando a sintaxe abaixo:

```
<xsl:for-each select = "padrão XSL">
    <!—Costrução do resultado-->
  </xsl:for-each>
```

Apesar da comparação com XML-QL, existe características desta linguagem que não são atualmente cobertas por XQL como variáveis, identificadores de objeto, expressões regulares em padrões, integração de dados de múltiplas fontes XML. O fato da linguagem não suportar variáveis, impede que sejam feitas consultas com *joins*, reduzindo o poder de expressão da linguagem.

5.4 WebSQL

Segundo [ARO 97], *WebSQL* propõe um modelo para a Web, como um banco de dados composto por duas relações (virtuais): *Document* e *Anchor* (Figura 28). *Document* é uma tupla do documento na Web e *Anchor* tem uma tupla para cada documento na Web. Esta relação permite o uso de uma linguagem similar a SQL, e quando esta relação é real usa-se a SQL, para realização das consultas. A pesquisa abaixo, mostra a dupla de URLs que possuem o mesmo título.

```
SELECT  d1.url,  d2.url
FROM  Document d1,  Document d2
WHERE  d1.title =  d2.title
    AND NOT  (d1.url  =  d2.url)
```

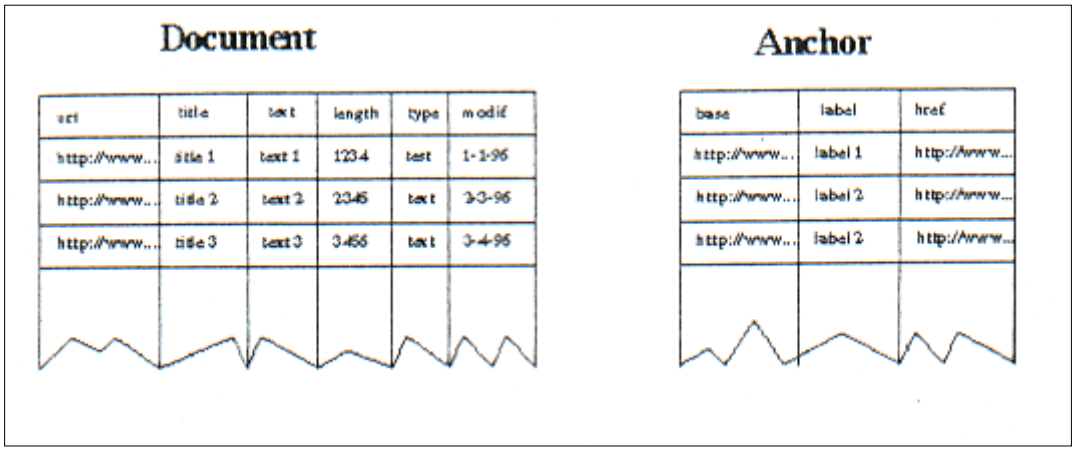

FIGURA 28. A Web de acordo com a linguagem WebSQL.

Porém se a relação entre o *Document* e *Anchor* for completamente virtual e não existir como enumerar todos os documentos da Web, nem como operar sobre eles, é necessário trabalhar diretamente sobre a relação virtual, materializando porções de documento, que após, poderão ser manipulados como uma relação convencional.

A consulta SQL abaixo, semelhante a anterior, porém acrescido apenas, do termo "SUCH THAT.... MENTIONS...., para cada variável, materializa uma porção da Web contendo todos os documentos que contém alguma palavra do termo mencionado. Este tipo de implementação existe, atualmente nos browsers Alta Vista, Excite, Hotbot entre outros.

```
SELECT  d1.url,  d2.rll
FROM Document  d1  SUCH THAT d1 MENTIONS   "something interesting",
        Document  d2  SUCH THAT d2 MENTIONS   "something interesting"
WHERE   d1.title  =  d2.title  AND NOT (d1.orl = d2.url)
```

5.5 WebOQL

Segundo [ARO 98], esta linguagem descreve ou trata as páginas da Web como *objetos atômicos* com duas propriedades: (1) as páginas contém ou não certo modelo padrão de texto, e eles separam os outros objetos, (2) proporciona acesso a estrutura interna das páginas para serem manipuladas, possibilitando assim, a criação de novas e complexas estruturas como resultado da pesquisa e, como os dados na Web são normalmente semi-estruturados, a linguagem enfatiza o suporte a esta característica.

A estrutura utilizada pela linguagem é chamada de ***Hypertree,*** que é organizada em *arcos rotulados,* que podem ser internos e externos. Os arcos internos são usados para representar as estruturas dos objetos e os arcos externos são usados para representar as referências (tipicamente hiperlinks) entre os objetos. Os arcos são chamados de *records* (registros).

A Figura 29 mostra uma Hypertree contendo descrições de publicações de um grupo de pesquisa, e a Figura 30 mostra a mesma aplicação usando um XML Wrapper genérico.

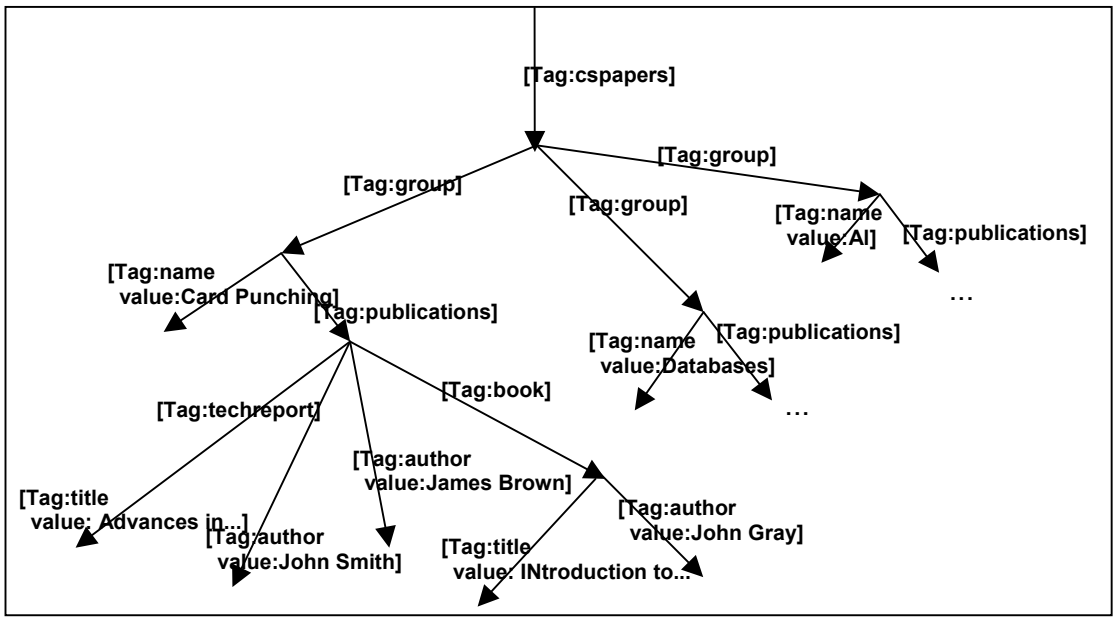

FIGURA 29. Hypertree das publicações de um grupo de pesquisa.
Fonte: [ARO98].

```
<?xml version="1.0"?>
<cspapers>
 <group>
  <name>Card Punching</name>
  <publications>
    <techreport>
      <title>Advances in Card Punching</title>
      <author>John Smith</author>
      <author>James Brown</author>
    </techreport>
    <book>
      <title>Introduction to Card Punching</title>
      <author>John Gray</author>
    </book>
 </publications>
</group>

 <group>
   <name>Databases</name>
   <publications>
     ...
   </publications>
 </group>
 ...
</cspapers>
```

FIGURA 30. Arquivo XML da aplicação.

WebOQL é uma linguagem funcional, mas as consultas são estruturadas usando-se cláusulas usuais como select, from, where form. Por exemplo, supondo-se que o arquivo "pubs.xml" contém as publicações mostradas na Figura 30, e deseja-se extrair todas as publicações de autoria de "Smith":

```
[tag: "result" /
select y
from x in browse {"file:pubs.xml") via ^* tag = "publications"),
                  y in x',
                  z in y'
where z.tag = "author" and z.value ~ "Smith"
]
```

Nesta consulta, x faz a interação entre as *subtrees* de "publications" e de "csP*apers*", atribuindo o valor para *x*, já a variável *y* interage na árvore simples do *x*, e, finalmente o resultado dessa consulta é "pendurado" em um novo arco chamado pela tag "*result*".

A Hypertree acima pode gerar uma nova Hypertree, mas geralmente uma consulta é uma função de um outro mapa ou Hypertree na Web. Por exemplo, seguindo este raciocínio citamos a criação de uma nova página para cada grupo de pesquisa, contendo apenas as publicações de cada grupo. A cláusula "as" especifica o nome do grupo que é usado para criar a URL de cada nova página criada.

5.6 XQuery

. Segundo [ROS 98], XQuery é uma linguagem bastante diferente das demais linguagens, uma vez que não utiliza, como as demais, os princípios de uma linguagem relacional de consulta como SQL, modificada pelo acréscimo de algumas poucas construções. Este problema surgiu porque o princípio dos bancos de dados relacionais, não é mantido nos documentos XML e, entre eles podemos citar:

- os dados são organizados em três níveis hierárquicos: tabela, registro e campo;
- a ordem dos registros(records) na tabela, não fazem parte da informação, e podem ser descartados ou modificados a qualquer momento;
- os campos não possuem uma estrutura interna que permita ao banco de dados conhecer: eles são unidades atômicas, e possuem tipos de dados, como integer (inteiro), date (data) ou string;
- as strings podem ocorrer em campos separados;
- não existe noção da distância entre os objetos de dados;
- o contexto ou a localização dos dados geralmente não são significantes (relatam o resultado da ordem e hierarquia): uma tupla idêntica é gerada se for copiada, o que pode ser representado apenas em documentos XML e não em documentos em linguagem natural;
- para ligar uma ou mais partes dos dados, não é usado nenhuma parte do "*schema*" para representar os dados.

As linguagens de consulta para dados são desenvolvidas trazendo a vantagem de apresentarem essas características acima citadas. Entretanto, exatamente por esta razão as consultas em XML não são perfeitas, onde os dados são completamente hierárquicos, ordenados e totalmente preenchidos com os objetos agregados a sua estrutura interna, utilizando sempre a linguagem humana para representar os objetos.

Segundo o autor, a linguagem é diretamente baseada em *XPointer*, por razões como:

- ser possível obter a consulta desejada utilizando poucas palavras e uma certa sintaxe;
- XPointer foi desenvolvido especificamente para XML, e contém determinados operadores de acesso navegacionais que são requeridos para o procedimento de ordenação das árvores, mas existe deficiência de encontrarmos outros modelos com a mesma finalidade;
- foi considerado seguro e robusto por um longo período, não sendo amplamente conhecido, mas é um modelo fechado do *Text Encoding Iniciative (TEI*), amplamente implementado;
- *XPointer* é um pequeno passo dos apontadores estendidos do TEI, justificados pela boa forma com que trabalham,, onde foram trocadas as marcas de pontuação por espaços, e a eliminação de algumas construções;
- tem sido utilizada em aplicações , não apresentado nenhuma lacuna ou deficiência significante na finalidade proposta.

Xquery consiste numa seqüência de passos, usando termos de locação (*location terms*) da *Xpointer*, onde cada passo seleciona nodos a partir do elemento raiz do documento que está sendo consultado.

No exemplo abaixo, no documento consultado é buscado todos os elementos FOOTNOTE, no qual o tipo do atributo tem valor 'CITATION'.

Descendant (FOOTNOTE & TYPE='CITATION').(REF)

5.7 XML-QL

Segundo [DEU 99], que propõe a XML-QL – XML Query Language, esta é uma linguagem que utiliza como base conceitos já desenvolvidos em linguagens para dados semi-estruturados. De acordo com o autor, "XML-QL pode expressar *consultas*, as quais extraem *porções* de dados de documentos XML, bem como pode fazer *transformações*, como por exemplo mapear dados XML entre DTDs, integrando dados XML de diferentes fontes".

Além disso a linguagem suporta operações *data-intensitive*, tais como *joins* e *agregações*, e tem suporte para construir novos documentos XML.

O modelo de dados proposto para XML é apresentado em duas versões: *ordenado* e *não ordenado*. O modelo *não-ordenado* representa um documento XML através de grafo, em que cada nodo é representado por uma string única, chamado *object identifier – OID* (identificador de objeto).

Na Figura 19, grafo XML não ordenado resultante da interpretação da DTD da Figura 16, onde as setas são rotuladas com as tags, os nodos com conjuntos de pares de atributo e valor e, as folhas com uma string de valor. Todo o gráfico possui um *nodo* distinto chamado *raiz*.

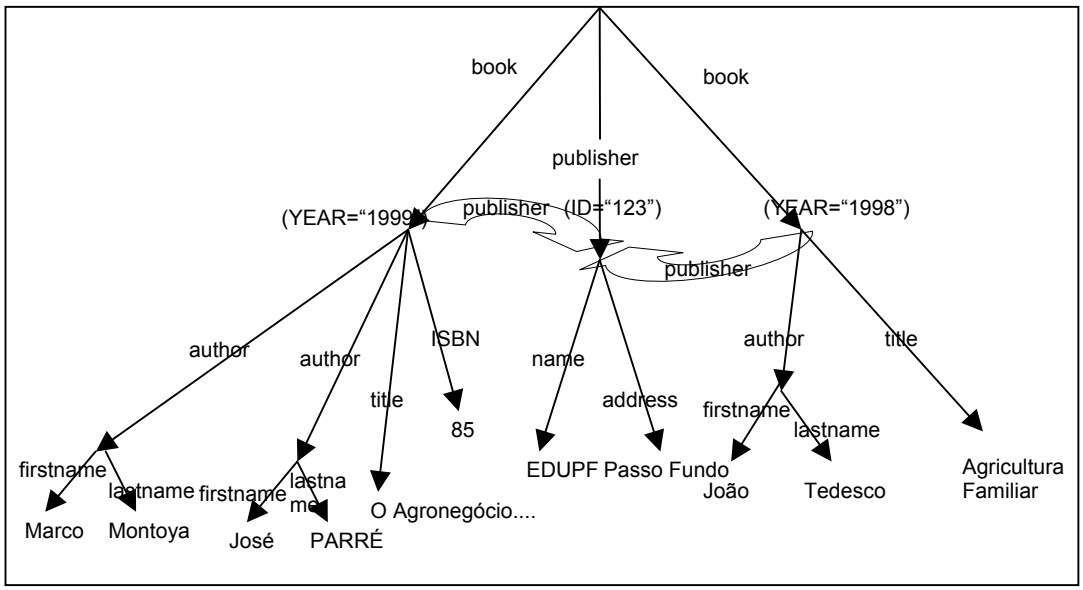

FIGURA 31. Grafo XML não ordenado da DTD da Figura 16.
Fonte: [KAD 99].

O modelo de dados estruturado é, em essência, semelhante ao ordenado, com a diferença que seus elementos seguem um ordenamento total, o que pode gerar a ocorrência de mais de uma seta partindo de um mesmo nodo, e uma maior complexidade nas consultas, em função das restrições de ordenamento. A Figura 20, mostra o grafo ordenado para a DTD da Figura 16, observando-se que cada *nodo* possui um número, que indica a sua ordem frente aos demais elementos. Para fazer essa ordenação são utilizados, quando existem os atributos ID, quando não existem é criado um identificador de objeto para cada elemento, por meio dos atributos do tipo IDREF e IDREFS, o que permite que sejam referenciados por outros elementos, permitindo assim consultas cruzadas.

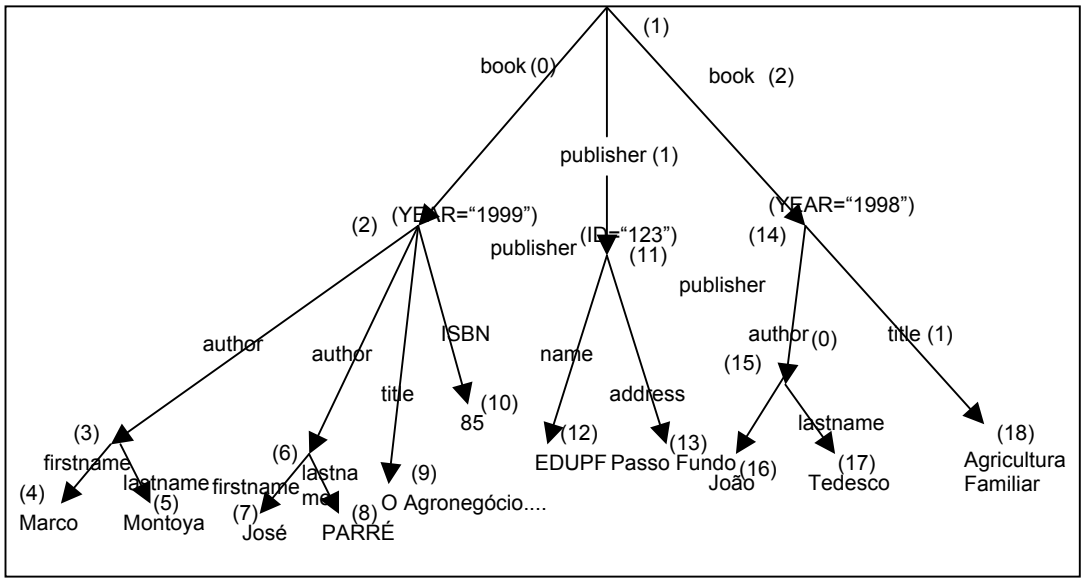

FIGURA 32. Grafo XML ordenado da DTD da Figura 16.
Fonte: Extraído de [KAD 99], baseado em [DEU99].

As consultas XML-QL podem recuperar dados XML ou construir novos documentos. O formato geral de uma consulta é WHERE *seleção* IN *origem* CONSTRUCT *resultado*, onde *seleção* são os dados a serem buscados nos documentos XML, *origem* é o endereço dos documentos e, *resultado* é o documento XML construído como resultado da consulta, baseando-se em *padrões de elementos* para buscar dados em um documento XML.

A consulta abaixo produz como resultado uma lista composta por todos os títulos e ISBN dos livros cujo sobrenome do autor seja "Silva", armazenados em um documento localizado no endereço *www.a.b.c/bib.xml*. Neste exemplo as variáveis *t* e *i* são relacionadas a cada par composto por título e número do ISBN, e como resultado, é produzida uma lista de autores ligados a *a* e de números de ISBN ligados a *i*.

```
WHERE     <book>
          <author><lastname>Silva</></>
                    <title>     $t  </>
                    <ISBN>   $i  </>
          </>  IN "www.a.b.c/bib.xml
CONSTRUCT <result>
                    <title>     $t  </>
                    <ISBN>   $i  </>
          </>
```

As consultas também podem ser aninhadas, a fim de agrupar elementos dentro de um contexto específico, como a consulta abaixo, na qual o resultado é uma lista contendo os títulos dos livros, cada um contendo a lista de seus autores. Também é utilizado um novo contexto, o CONTENT_AS, que liga o conteúdo do elemento combinado com a variável $p, que pode na seqüência ser usada para outras consultas. As variáveis $f e $l, são utilizadas para mapear o primeiro nome e o sobrenome dos autores, tanto para elementos <article> quanto para elementos <book>, o que faz com que haja a combinação de ambos, produzindo como resultado uma lista de artigos.

```
WHERE     <article>
                    <author>
                              <firstname>   $f </>
                              <lastname>   $l </>
                    </>
          </>   CONTENT_AS  $a IN "www.a.b.c/bib.xml
          <book>
                    <author>
                              <firstname>   $f </>
                              <lastname>   $l </>
                    </>
          </>   IN "www.a.b.c/bib.xml
CONSTRUCT <article>   $a  </>
```

6 Ontologias

6.1 Conceito e Características

O termo "Ontologia" já é conhecido e aplicado a bastante tempo nas áreas da Filosofia e da Epistemiologia, significando, respectivamente, um "sujeito de existência" e um "conhecimento ou saber". Na década de 90, este conceito passou a ser utilizado na área de IA, para descrever conceitos e relacionamentos utilizados por um agente ou comunidade de agentes (conhecimento compartilhado), [GRU93] citado por [MEL00].

A definição de Ontologia dentro da Ciência da Computação ainda não está consolidada, porém vem sofrendo aprimoramentos a medida que desperta o interesse de um número crescente de pesquisadores da área.

Um modelo conceitual é um modelo que descreve, dentre outras coisas, a estrutura do banco de dados em um alto nível de abstração. Uma Ontologia não representa a estrutura das fontes de dados associadas a ela, apenas propõe uma estrutura de consenso para conceitos e relações que são úteis para grupos de usuários, sendo essa estrutura instanciada pelo BDSE através de traduções realizadas por *wrappers* e mediadores.

Assim, uma Ontologia é um mecanismo de interpretação parcial ou total do universo de dados semi-estruturados de uma ou mais fontes, não existindo obrigatoriamente uma correspondência direta entre possíveis estruturas implícitas dessas fontes e a estrutura da ontologia. A Ontologia é definida então, não com a finalidade de definir a estrutura de um BDSE e sim visando definir um vocabulário de trabalho para um grupo de usuários.

Outro aspecto importante associado a ontologias, bastante enfatizado em termos de pesquisa, é o *reuso* de conhecimento definido em Ontologias já existentes. Esse fator afeta a desenvolvimento de ferramentas e metodologias, exigindo que existam *mecanismos de tradução* entre formalismos de representação e a definição de *níveis de reuso*, que sugerem que Ontologias estejam organizadas em módulos de conhecimento, que especificam níveis de detalhamento deste conhecimento.

Neste sentido, é interessante desenvolver Ontologias amplas, contendo conhecimento de "senso comum" e uma capacidade de aumentar esse conhecimento através da recuperação de fontes de dados on-line.

Segundo [SIL99], uma Ontologia pode ser definida como um conjunto de conceitos de interesse em determinado domínio de aplicação, juntamente com os relacionamentos entre estes conceitos. Ontologias fornecem uma descrição concisa, uniforme e declarativa das informações semânticas, independente do local e da forma como os dados estão armazenados.

O mesmo autor menciona várias abordagens de aplicação de Ontologias sobre dados oriundos da Web, sobre dados textuais ou sobre dados semi-estruturados e sobre sistemas de informações globais.

A partir destas abordagens pode-se aplicar, XML, com a constituição de uma camada semântica sobre os repositórios de dados descritos sobre determinada Ontologia, sobre a qual os usuários executam consultas, ficando o sistema com a responsabilidade de gerenciar o acesso aos dados.

Segundo [ERD00], XML atualmente é o caminho ou direção que domina a Web, e muitas aplicações estão sendo e serão desenvolvidas utilizando esta nova linguagem. XML é destinada a representar definição de tipos de documentos e para domínios com propósitos pré-determinados, como por exemplos apresentações multimídia, páginas HTML de conteúdos arbitrários, transações de negócios (XML/EDI).

Os documentos XML explicitam a estrutura textual do documentos que poderão serem facilmente acessados pelas aplicações, por interfaces padrões como SAX e DOM. As tags têm propósito semântico, ao contrário das tags em HTML que tem apenas o objetivo de definir o layout, podendo assim serem exploradas para diferentes finalidades.

Segundo este autor, Ontologia é uma especificação formal da conceitualização e proporciona uma semântica básica através do processamento do documento XML.

Em um resumo realizado por [MEL00], coletando partes das diferentes definições encontradas nas bibliografias, definiu Ontologia sob o ponto de vista de banco de dados como sendo *"uma especificação parcial de um domínio da realidade, que descreve conceitos, relações entre eles e regras de integridade. A idéia de Ontologia como modelo e meta-modelo sugere que diversos níveis ontológicos podem existir e se relacionar."*

Existem algumas características fundamentais que devem ser levadas em consideração na construção de uma ontologia:

a) aberta e dinâmica – para adaptar-se às mudanças do domínio associado, devendo ser o mais automatizada possível;
b) escalável e interoperável – deve ser facilmente escalável para um amplo domínio e adaptável a novos requisitos, devendo para isso ser simples;
c) de fácil manutenção – deve ser ao mesmo tempo dinâmica e de fácil manutenção por especialistas humanos;
d) semanticamente consistente – deve manter conceitos e relacionamentos coerentes;
e) independente de contexto – a ontologia não deve conter termos muito específicos em um certo contexto, porque lida com fontes de dados de larga escala.

6.2 Extração de dados baseada em Ontologias

Aplicada a dados WEB que possuem um cerro número de constantes (strings) identificáveis em seus documentos e, cujos conceitos do seu domínio podem ser descritos em uma ontologia relativamente pequena, que serve de base para o reconhecimento e extração de informações, [MEL00].

Alguns exemplos de domínio de pequena escala que podem ser citados são dados de obituários, dados de monografias de conclusão de curso.

Os passos seguidos nesta abordagem, a extração baseada em Ontologias são os seguintes:

1º) Definir um modelo ontológico do domínio, ou especificação da ontologia:

- Ontologia para o domínio de monografias de conclusão de curso da URCAMP/Bagé (estudo de caso em questão):

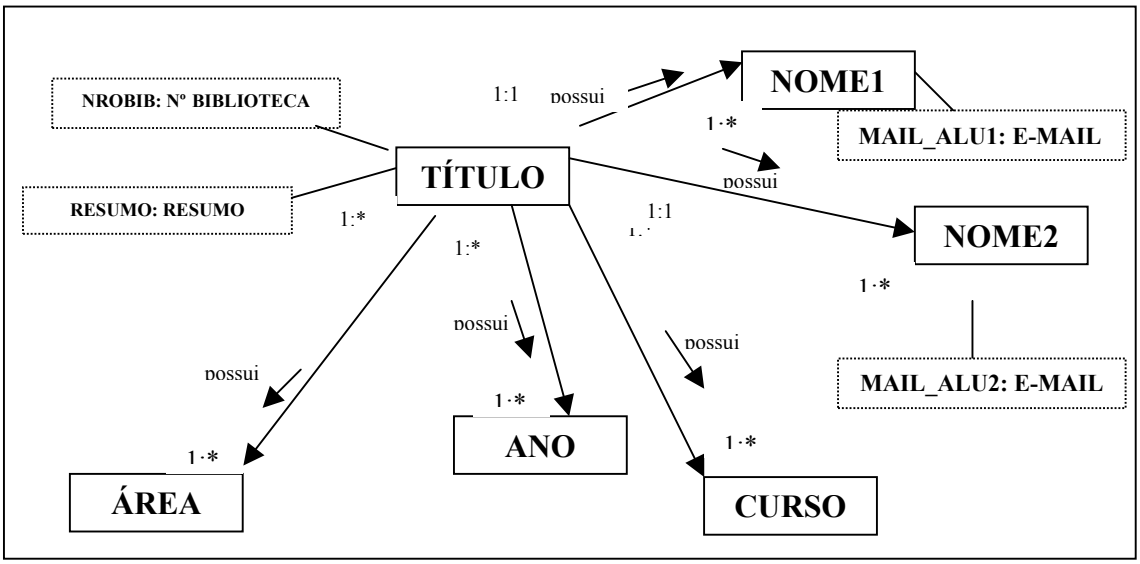

FIGURA 33. Ontologia para o domínio da aplicação.

2º) Geração do esquema do BD

É uma especificação textual da ontologia definida, servindo de entrada para o parser, resultando nos:

- FDDs (frames de descrição de dados), que encapsulam as propriedades de conceitos, descrevendo regras e padrões léxicos;
- especificação do esquema de BD.

3º) Extração de registros não estruturados

Investigar as páginas Web, através de um extrator de dados que separa a informação dentro da página em porções de dados (registros) que representam instâncias de conceitos na ontologia, seguindo o seguinte algoritmo:

1) montar uma árvore cuja hierarquia representa o aninhamento de tags HTML(raiz padrão), chamada árvore de tags;
2) buscar o nodo com a maior subárvore, ou maior número de filhos, representando a estrutura principal do documento;
3) para cada subárvore deste nodo, conta-se o número de ocorrências de cada filho , concentrando-se em tags dominantes(TDs), ou seja tags com alta frequência;
4) aplica-se a heurística da maior ocorrência: se existe apenas uma TD é selecionada como separadora de registros, porém se existe mais de uma verifica-se o número de ocorrências escolhendo-se a que tiver menor desvio padrão;
5) após encontrada a tag separadora, modifica-se o documento inserindo-se a tag "######" após cada ocorrência da TS e remove-se as tags do documento., tendo-se assim os dados de interesse separados.

4º e 5º) reconhecimento de conceitos e relacionamentos, e geração de instâncias para o esquema do BD

Para cada registro de interesse (ou registro não estruturado) no documento realizam-se 2 passos:

1) produzir uma tabela do tipo descritor/string/posição (TAB), formada por constantes e palavras chaves reconhecidas no registro de interesse;
2) Casam-se atributos do esquema do BD com valores de TAB, gerando-se instâncias.

Cada entrada em TAB descreve uma constante ou palavra chave. O primeiro campo, descritor, se for constante mantém o nome do conjunto de objetos ao qual a palavra chave se aplica. O segundo campo, string, guarda a constante ou palavra chave, com as possíveis transformações realizadas pelas regras dos frames de descrição de dados.

6.3 Aplicações

Na área de IA(Inteligência Artificial), as Ontologias têm sido aplicadas há vários anos e, mais recentemente, a área de sistemas de RI emprega também ontologias como provedores de semântica para o vocabulário de certas classes de documentos, detectando melhor a relevância de documentos em busca de termos que são palavras-chave

Em Banco de Dados, têm sido aplicadas principalmente em banco de dados heterogêneos e *Data Warehouses* como modelos conceituais globais, resultantes de uma concordância na definição de entidades e relacionamentos.

Já em Sistemas de Informação, tem a finalidade de manutenção de conceitos, em um alto nível de abstração (como modelos ou meta-modelos), que sejam úteis para várias aplicações

Considerando BDSE, que trabalham com fontes de dados heterogêneas, as ontologias são importantes no sentido de manter esquemas de conceitos relevantes da realidade, e suas relações, de modo a guiar, por exemplo, a extração e a estruturação dos dados.

Atualmente uma outra área, ou uma outra tendência de aplicação de ontologias está sendo a área de comércio eletrônico, que brevemente se tornará o espaço dominante para a maioria das atividades de negócio, sendo necessária uma padronização dos modelos de negócio, processos e arquiteturas destes sistemas. Essa tarefa não é muito fácil, pois as práticas comerciais variam muito devido a razões técnicas e políticas. Uma solução para esse problema é o uso de Ontologias compartilhadas como base para a interoperação entre parceiros de negócios em mercados eletrônicos. O uso de Ontologias aceleram a penetração do CE dentro dos variados setores da sociedade, reduzindo a necessidade de uma padronização muito rígida que poderia ser um fator limitante.

Nesse sentido, uma tendência de padronização é o uso da linguagem XML para uma representação sintática (através de tags especiais) de conhecimento específico de CE.

As Ontologias poderiam ser vinculadas a especificações XML para prover o suporte semântico.

6.4 Aplicação de Ontologias a BDSE

A aplicação de Ontologias a Banco de Dados Semi-Estruturados (BDSE) é motivada pelos seguintes fatores:

- poucos trabalhos desenvolvidos até hoje;
- as máquinas de RI e técnicas de *parse* de linguagem natural são limitadas no entendimento semântico das fontes de dados;
- exigem documentos com uma estrutura uniforme, conhecida pelo usuário, pois a pesquisa é feita por padrão e por casamento de estruturas.
- requerem editores e analisadores especializados para interpretação semântica (metadados como anotações em linguagem de marcação).

As Ontologias dão suporte a consultas declarativas no estilo de consultas a banco de dados, a conceitos e relações do domínio, evitando expressões de navegação em estruturas complexas de dados semi-estruturados.

Provêem uma camada semântica sobre fontes de dados heterogêneas, melhorando a interoperação entre estas fontes, podendo-se até mesmo mapear e combinar dos dados extraídos.

Existem várias propostas para aplicação de Ontologias a BDSEs, entre elas:

6.4.1 O Sistema de Informação Global Observer

Esses sistemas de informação global(SIG), têm como característica o gerenciamento de repositórios de dados distribuídos e altamente heterogêneos, como é o caso da Web.

Observer é um sistema de informação global em desenvolvimento, que utiliza Ontologias para o processamento de consultas.

As Ontologias neste caso, são consideradas metadados, organizados em uma hierarquia de especialização de termos (conceitos e regras), que capturam a semântica do conteúdo dos dados de repositórios, abstraindo heterogeneidade e distribuição de dados.

Os usuários formulam consultas diretamente sobre uma ontologia selecionada, podendo inclusive definir suas próprias ontologias , sem que os termos definidos estejam necessariamente presentes em algum repositório de dados.

6.4.2 Linguagem SHOE

SHOE (Simple HTML Ontology Extensions) é uma linguagem desenvolvida nos Estados Unidos que estende o HTML com tags orientadas a conhecimento, provendo uma estrutura para aquisição deste conhecimento, citado por [MEL00].

Taxionomias e regras de inferência disponíveis em Ontologias que são referidas em SHOE habilitam a descoberta de conhecimento implícito. Foi desenvolvida, a título de experiência, para um domínio veterinário, mais especificamente, para aplicações que lidam com informações referentes a doenças que afetam o gado bovino (sistemas DGB)

6.4.3 ONTOBROKER

É uma ferramenta desenvolvida na Alemanha, que processa descrições de conteúdo em HTML, XML e RDF, provendo recuperação inteligente da informação. Sua arquitetura é formada por quatro componentes que interagem com a ontologia:

a) máquinas de consulta – recebe as consultas e as responde, verificando o conteúdo de um banco de dados cujos dados são preenchidos pelo agente de informação e pela máquina de inferência. Atua sobre uma linguagem única para consulta e representação dos dados, que se baseia em uma expressão elementar de consulta do tipo:

$$X \in c \wedge attribute(x) = v,$$

A partir do qual expressões mais complexas são construídas. Essa linguagem é chamada de lógica de frames (formalismo lógico OO presente em alguns bancos de dados dedutivos). Sendo especificadas nesta linguagem, permite a especificação de hierarquias de herança múltipla de conceitos, relacionamentos, atributos e regras de derivação de novos conhecimentos. As especificações são mantidas no BD. A nível de usuário final, uma interface gráfica com recursos de navegação em ontologias, permite a formulação de consultas interativas e a derivação de consultas textuais em lógica de frames, com apresentação de resultados tabulares.

b) agente de informação – responsável pela coleta de conhecimento factual na Web, utilizando vários estilos de meta-anotações diretas, como XML e HTML-A.
c) Máquina de Inferência- usa fatos, terminologia e os axiomas das ontologias para derivar conhecimento factual adicional.
d) Gerenciador de BD – estrutura o banco de dados.

Ao nível de usuário final, uma interface gráfica com recursos de navegação em ontologias, permite a formulação de consultas interativas e a derivação de consultas textuais em lógica de frames, com apresentação de resultados tabulares.

O agente de informação extrai conhecimento de 4 formas:

HTML-A → integra anotações semânticas em documentos HTML através de um browser para coletar páginas, extrair anotações, e convertê-las (parse) para um formato chamado HTML-A, que fica armazenado no banco de dados.
Wrappers → acessam fontes de dados que não utilizam as linguagens de anotação da ferramenta, como alguns documentos HTML que possuem alguma estrutura.
RDF → analisa anotações RDF (anotações semânticas a documentos que não levam em conta a sua estrutura), utilizando um parser que traduz suas descrições em triplas que são mantidas no banco de dados.
DTDs XML → DTDs gerados a partir da ontologia associam semântica a documentos XML

O agente de informação é responsável pela manutenção do BD da ferramenta, coletando periodicamente informações da Web e faz cache da mesma, e a máquina de inferência usa essa cache para responder consultas, processo semelhantes as máquinas de RI.

6.4.3.1 Modelagem Conceitual de Documentos XML

XML permite a inserção de novas tags com propósito semântico que podem ser explorados para as mais diversas finalidades.

O agente de informação de *Ontobroker* mapeia conceitos e atributos de ontologias para elementos de uma estrutura XML através de DTDs. As ontologias funcionam como unificadoras da sintaxe de documentos XML e propiciam consultas semânticas sobre os mesmos, que abstraem a sua estrutura.

Esta tarefa de mapeamento é feita por um módulo do agente de informação chamado *DTDMaker* , que recebe especificações ontológicas em lógica de frames e gera arquivos DTD XML. Estes DTDs XML definem classes de conceitos que auxiliarão na marcação de documentos XML, a tradução é feita da seguinte maneira:

Conceitos de ontologia → **tipos elemento** no DTD

Atributos de conceito da ontologia → **subelemento** no DTD

Atributos que representam **relacionamentos** na ontologia → subelementos que tem como **tipo** o **elemento** correspondente ao conceito relacionado , conteúdo do tipo atômico string PCDATA (padrão em DTDs).

A **herança** é especifica em DTDs através das chamadas **entidades de parâmetro** que definem strings que podem substituir uma entidade ao longo da DTD. Cada string representa um subconceito.

A seguir é mostrada a especificação parcial de uma Ontologia em lógica de frame, referente ao estudo de caso deste trabalho. (Figura 21).

```
Object [ ].
Aluno::Object.
                Nome1::Aluno.
                Nome2::Aluno.
Mail-alu1::Aluno.
                Mail-alu2::Aluno.
Curso::Object.
Ano::Object.
Tit::Object.
        Area::Tit.
        Nrobib::Tit.
        Orient::Tit.
        Resumo::TiT.
Aluno[nome1=>>string;nome2=>>string;mail-alu1=>>string;mail-alu2
=>>string;].
Curso[curso=>>string].
Ano[ano=>>string};
Tit[area=>>string;nrobib=>>string;orient=>>string;resumo=>>string].
....
FORALL Aluno, Tit Tit: Nrobib[nome->>nome2....
.....
```

FIGURA 34. Especificação parcial em lógica de frame da aplicação.

6.4.4 Proposta de Embley

Estratégia interessante para classes de documentos de certos domínios, como anúncios classificados, informações turísticas , que representam constantes facilmente identificáveis (ricos em dados) e seguem um certo padrão de discurso.

6.4.5 O sistema InfoSleuth

É resultado de um projeto de visa desenvolver soluções para os problemas de acesso e manipulação de dados heterogêneos em um ambiente distribuído com a Internet. Utiliza uma abordagem baseada em agentes cooperativos, cada um com uma especialidade, para obtenção e análise de informação em diversos níveis de abstração.

Além dos agentes, o sistema conta ainda com várias Ontologias, que descrevem conceitos de domínios de aplicação, análises e eventos.

6.4.6 MOMIS

MOMIS (*Mediator envirOnment for Multiple Information Sources*) é um ambiente para integração e consulta a bancos de dados estruturados e semi-estruturados, em desenvolvimento na Universidade de Modena, na Itália. Utiliza uma abordagem baseada num modelo conceitual para integração de informação de fontes de dados cuja arquitetura tem as seguintes características:

- um *modelo conceitual ontológico* OO especificado na linguagem ODL $_I{}^3$, uma linguagem padrão de mediação do grupo I^3 / POB, que introduz construtores para suportação integração semântica;
- *wrappers* para tradução de esquemas em representação ODL $_I{}^3$;
- componentes *mediador* e *processador de consultas*, baseado em duas ferramentas já disponíveis ARTEMIS e ODB-Tools.

A base de informação do sistema é um *Thesaurus* que serve como uma *ontologia compartilhada* para fontes de dados.

6.4.7 WebKB

É uma ferramenta para indexação, anotação e exploração de conhecimento em documentos da Web, ainda em desenvolvimento na Austrália.

Utiliza uma linguagem geral para Representação do Conhecimento, baseada no formalismo de *grafos conceituais*, que são notações gráficas lineares que definem associações e especializações entre termos formais predefinidos em uma ampla ontologia.

WebKB define operadores aplicados aos grafos conceituais e estende a sua notação para outras formas de representação do conhecimento, como estruturas HTML, textos indentados e inglês formalizado.

Sua arquitetura apresenta três componentes:

- uma máquina de pesquisa, para busca de texto/conhecimento, que pode gerar novos documentos;
- uma interface de pesquisa de texto/conhecimento em HTML, incluindo um editor de conhecimento na linguagem da ferramenta;
- uma ontologia única;

A Ontologia foi projetada para facilitar a criação de Ontologias de aplicação e a construção de comandos de conhecimento. É uma extensão do BD léxico WordNet, possuindo em torno de 200 tipos de relações básicas entre conceitos, classificados em relações temáticas, matemáticas, espaciais, temporais, retóricas e argumentativas. Também define próximo de 200 tipos de conceitos gerias, necessários para representar conhecimento e assinaturas de tipos relações.

A existência desta ontologia permite categorizar semanticamente objetos criados por usuários. Também utiliza abordagens léxicas, estrutural e baseada em conhecimento na definição de comandos de consulta e de definição de conhecimento.

A abordagem léxica é utilizada para processamento de documentos da Web através de um comando estilo *Unix* chamado *acessibleDocFrom*, que lista documentos acessíveis direta ou indiretamente a partir de um certo número de links.

As abordagens estrutural e baseada em conhecimento são empregadas na linguagem geral para RC. Através de tags especiais HTML (<KR>...<KR>) ou entre os símbolos "$(" e ")$", embutem-se os comandos de especificação, indexação ou consulta s conhecimento em documentos da Web.

Os termos da Ontologia são utilizados nestes comandos para definir ocorrências de relações ou conceitos. Esta estratégia procura aproximar a definição do conhecimento da fonte de dados em que ele se encontra.

7 XML e PHP

A PHP oferece suporte a um conjunto de funções que podem ser usadas para escrever aplicativos XML baseados em PHP. Essas funções podem ser usadas para analisar documentos XML bem formados. O analisador XML, em PHP, é um analisador baseado em fluxos. O documento XML é levado ao analisador em seções e, quando o analisador analisa o documento e reconhece os nós, ele chama a manipuladora registrada adequada ou apropriada, não verificando a validade do documento XML.

A extensão XMP PHP oferece suporte ao conjunto de caracteres Unicode através de duas codificações de caractere diferentes, codificação de origem e codificação de destino.

A codificação de origem é realizada quando o documento XML é analisado, sendo que a padrão usada pelo PHP é a ISO-8859-1. A codificação de destino é executada quando a PHP passa os dados para as funções manipuladoras registradas, afetando os dados de caractere, assim como os nomes de tag e destinos de instrução de processamento.

7.1 A API XML em PHP

Um script PHP que analisa um documento XML deve realizar as seguintes operações:

- criar um analisador XML;
- registar funções manipuladoras (funções de retorno) no analisador, e este por sua vez, chamará essas manipuladoras registradas à medida que e quando reconhecer diferentes nós no documento XML que está sendo analisado;
- ler os dados do arquivo XML e passá-los ao analisador;
- liberar o analisador depois que o arquivo tenha sido analisado por completo.

A seguir é apresentado um analisador XML, com comentários, extremamente simples, que exibe as tags de abertura e fechamento e os dados de caractere.. Inicialmente define as funções manipuladoras que informam o analisador que ação executar ao encontrar um tipo de nó específico.

.

```
<?php
//definição da manipuladora de tags de abertura de elemento
function startElementHandler($parser, $name, $attribs) {
echo("&lt;$name$gt;<BR>");
}
//definição da manipuladora de tags de fechamento de elemento
function endElementHandler($parser, $name) {
echo("&lt;$name$gt;<BR>");
}
//definição da manipuladora de tags de  caractere
function cdataHandler($parser, $name) {
echo("$data<BR>");
}
//criação do analisador
$parser=xml_parser_create();
//registra as manipuladoras de elemento inicial e final
```

```
xml_set_element_handler($parser,"startElementHandler","endElementHandler");
//registra o analisador de dados de caractere
xml_set_character_data_handler($parser,"cdataHanldler");
//abre o arquivo XML
if (!($fp = fopen($file, "r"))) {
  die("could not open $file for reading");
}
//lê trechos de 4kb e passa para o analisador
while ($data = fread($fp, 4096)) {
  if(!xml_parse($parser, $data, feof($fp))) {
   die(sprint("XML error %d %d, xml_get_currentline_number($parser),
              xml_get_current_column_number($parser)));
   }
}
?>
```

O arquivo **test.xml**, seria estruturado da seguinte forma:

```
<?xml version="1.0"?>
<books>
    <book>Pro PHP</book>
    <book>XML in IES</book>
    <book>Pro XML</book>
</books>
```

A Figura 35, mostra o que será exibido na janela do navegador:

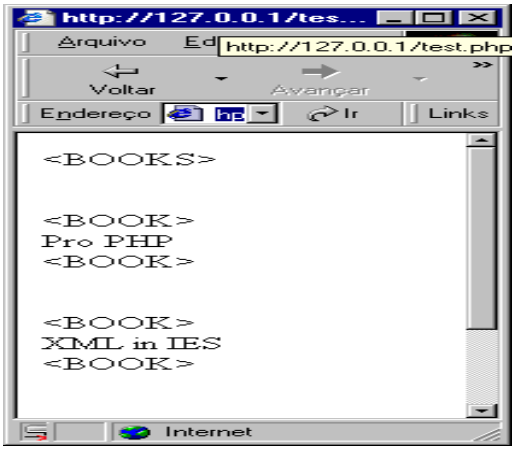

FIGURA 35. Resultado de um script em PHP, manipulando o arquivo TEST.XML

Antes de podermos analisar um documento XML, precisamos escrever funções que tratem dos vários nós do documento. Sendo necessário que escreva-se uma função que manipule os tags de abertura de elementos XML, outra que manipule os tags de fechamento, sendo necessário também que se atribua manipuladoras para dados de caracteres, para instruções de processamento, entre outras. Essas manipuladoras devem ser registradas no analisador XML.

Após criadas as manipuladoras, pode-se criar o analisador XML, propriamente dito, através da função *xml_parser-create( ),* que retorna um tipo de tratamento de acordo o resultado, *true* se obtiver êxito, valor inteiro positivo e *false* no caso de erro. Esse valor será passado como argumento para todas as chamadas de função de registram funções manipuladoras no analisador ou que alteram as opções do analisador.

A função *xml_set_default_handler( )* registra a manipuladora padrão no analisador XML. A manipuladora padrão é chamada para todos os nós do documento XML para os quais as manipuladoras não podem ser registradas (declaração de versão, declaração de DTD e comentários) e também para todos os outros nós para os quais os analisadores não são registrados no analisador.

A função xml_parse ( ) passa o conteúdo do documento XML para o analisador. que realiza uma análise real do documento, chamando as manipuladoras adequadas à medida que for encontrando os nós no documento.

7.2 Aplicativos XML PHP

XML está sendo usada como o formato para troca de dados entre diferente aplicativos comerciais, a DTD para o setor está sendo definida e PHP pode ser usada para fornecer a interface baseada na Web para esses aplicativos comerciais. PHP também pode ser usada para pesquisar documentos XML, como exemplo se todos os artigos em um site Web são escritos usando uma mesma DTD, que define elementos para autor, titulo, resumo, etc., então PHP pode ser usada para pesquisar artigos , dependendo do autor, do título, etc.

Atualmente existem poucos navegadores que tem suporte interno para PHP. A PHP também pode ser usada no lado do servidor para analisar documentos XML e retornar HTML pura para o cliente, permitindo assim, a todos os navegadores , independente do suporte ou não a XML, verem documentos XML.

O script PHP também pode enviar o documento XML ou HTML convertida para o cliente, dependendo do navegador que tivesse enviado o pedido. Por exemplo se o pedido de um arquivo XML fosse enviado do Internet Explorer 5 , então o script PHP poderia simplesmente retornar o arquivo XML, caso contrário, em outro navegador, o script converteria os dados XML para HTML e enviaria uma página HTML para o navegador. Sendo assim. PHP pode ser usada para apresentar diferentes visões do mesmo documento XML, excluindo ou modificando nós dentro do documento.

8 Descrição do Estudo de Caso

8.1 Introdução

O estudo de caso propõe o desenvolvimento de uma aplicação, utilizando XML e linguagens de consulta para XML, baseado em *Ontologias*, com o objetivo de permitir consultas e o armazenamento de informações referentes aos alunos concluintes do Curso de Informática e Curso Técnico em Informática, da Universidade da Região da Campanha – URCAMP/Bagé-RS, disponibilizando à Universidade, uma nova ferramenta, com informações referentes aos cursos em questão e, principalmente, utilizando uma nova tecnologia, que tende a tornar-se padrão na *WEB*.

A aplicação permite a visualização e a manipulação dos dados referentes, aos estágios de conclusão de curso dos alunos dos Cursos de Informática (Ensino Superior/3º grau) e Técnico em Informática (Ensino Médio), da URCAMP/Bagé-RS.

As informações armazenadas são referentes ao nome do aluno concluinte, e-mail do aluno, o curso, ano/semestre de conclusão do curso, tema ou título do trabalho/monografia ou estágio desenvolvido para a conclusão do referido curso, um pequeno resumo do trabalho desenvolvido, a nota final do estágio, informações relativas ao professor orientador, e toda e qualquer informação que se julgar relevante para a exibição completa dos dados, no momento do desenvolvimento da aplicação.

A proposta inicial, é que a aplicação permita a visualização das monografias/relatórios de estágio dos alunos em uma forma de relatório geral. Também é possível fazer consultas pelo nome do aluno, por área de estágio, por professor orientador, por ano ou semestre de conclusão do curso.

No desenvolvimento da aplicação, inicialmente, foi criada uma *DTD (Document Type Definition*), onde se define as *tags ou marcações,* de cada documento, quais destas *tags* poderão conter outras *tags*, sua seqüência, seus atributos e valores,

Após, foi criado o documento *XML*, utilizando o *XML Notepad* (Microsoft), para edição e manipulação dos dados existentes, em formato *HTML*.

O próximo passo foi a criação em *HTML*, de uma página que interpreta o documento *XML* e, faz a exibição dos dados em diferentes formatos e, as rotinas de consultas, que também podem ser executadas de diferentes formas.

Existem várias maneiras de se visualizar documentos *XML*, porém se o browser entende o padrão *XML*, o documento é enviado ou submetido diretamente ao browser, ou então pode-se usar uma folha de estilo para transformar o arquivo *XML* em algo que o *browser* "entenda" ou consiga interpretar..

No presente estudo de caso, optou-se pelo primeiro caso, utilizando-se o *HTML*, para exibir os dados, pois atualmente, a maioria dos browsers já conseguem interpretar *XML*.

Inicialmente a intenção era que a aplicação, faria as consultas a base de dados em *XML* utilizando a linguagem de consulta *XML-QL* e a técnica de

extração de dados, a *extração semântica de dados baseada em ontologias*, que segundo pesquisas, é mais precisa, facilita o entendimento da estrutura do documento, gerando um modelo mais rico, gerando como resultado, tabelas de banco de dados relacionais. Após o estudo aprofundado das diversas linguagens de consulta para XML, e várias tentativas optou-se por usar um script para desenvolver apenas a parte referente as consultas na aplicação.

A utilização de ontologias para representar o domínio do problema, se deve a dificuldade e a necessidade que existe em se conhecer a estrutura interna dos documentos, permitindo assim que as consultas sejam feitas mediante o uso dos conceitos da ontologia.

8.2 Descrição e Estrutura dos Cursos Envolvidos

A seguir é apresentado uma breve descrição de cada um dos cursos envolvidos no trabalho, quanto a estrutura curricular, tempo de existência do curso, número de alunos já diplomados, enfim, uma série de informações que têm o objetivo de facilitar o entendimento dos cursos como um todo.

8.2.1 Curso de Informática (Ensino Superior)

O curso de Informática da URCAMP/Bagé, formou até hoje em torno de 170 alunos desde a sua primeira turma em 1995 até o 2º semestre de 2000. A duração do curso é de 4 anos ou 2700 horas, incluindo o Estágio de Conclusão, com carga horária de 180 horas realizado no último semestre.

O estágio de conclusão é uma disciplina na qual o aluno desenvolve um trabalho prático, acompanhado de um relatório, dentro de uma área da Computação de seu maior interesse, utilizando para tal, os conhecimentos adquiridos ao longo do curso. Deverá estar inserido dentro das área de Computação Gráfica, Inteligência Artificial, Matemática Computacional, Projeto de Software / Sistema, Avaliação de Software / Sistema, Projeto / Avaliação de Hardware, Projeto / Avaliação / Otimização de Redes, Avaliação de Metodologia em Desenvolvimento, Prospecção Tecnológica e Levantamento (Survey) do Estado-da-Arte com Estudo de Caso.

Durante o período de estágio é realizado o Seminário em Andamento e, após é concluído o Relatório de Conclusão, que deverá ser apresentado e avaliado perante uma Banca Examinadora.

Após o término, uma cópia do relatório, revisada é enviada para a Biblioteca para ser cataloga e guardada, para posteriores consultas.

8.2.2 Curso Técnico de Informática (Ensino de Nível Médio)

O curso Técnico em Informática, existe desde 1992 e formou a sua primeira turma em 1995, inicialmente era chamado de Técnico em Processamento de Dados, a partir de 1999 com a base curricular reformulada passou a chamar-se de Curso Técnico em Informática.

Atualmente o curso possui 1500h, incluindo o Estágio Prático Profissionalizante de 240 horas (no mínimo), realizado dentro de alguma das áreas pré-definidas, como por exemplo, Hardware, Desenvolvimento de Sistemas, Análise de Sistemas, Redes, Editoração Eletrônica, Treinamento de Usuários e Desenvolvimento para Internet, que deverá ser realizado ao final do curso após cursar todas as disciplinas.

Este estágio, é realizado em empresas locais, sob orientação de um professor da Escola e supervisionado pela coordenação de estágios, num período variável de acordo com a disponibilidade da Empresa atingindo, no mínimo, um total de 240 horas.

Após o término das horas de Estágio na Empresa, o aluno tem um período de 30 dias para elaborar o Relatório de Conclusão de Curso, de acordo com as normas estabelecidas no Manual do Estagiário. Após, é realizada uma Banca Avaliadora, composta por três professores do curso e a Coordenação de Estágios, juntamente com o professor orientador, para avaliar e atribuir uma nota ao estagiário. Após, uma cópia do relatório é disponibilizada para a Escola.

8.3 Vantagens da Disponibilização na WEB

O objetivo principal da aplicação é permitir a disponibilização, visualização e a manipulação dos dados referentes, aos estágios de conclusão de curso dos alunos dos *Cursos de Informática* (Ensino Superior/3º grau) e *Técnico em Informática* (Ensino Médio), da URCAMP/Bagé-RS, utilizando para tal, uma nova tecnologia, o XML.

Além disso, também a disponibilização da base de dados em XML, possibilita que futuramente estas informações possam ser exibidas de uma forma diferente, ou até mesmo reaproveitadas, para uma outra pesquisa qualquer.

XML é um veículo excelente para se publicar dados na Web, que devido as habilidades referentes a apresentação, do HTML, juntas as duas linguagens de marcação se complementam. Diferentemente do HTML, XML é ideal para a criação de aplicativos Web complexos, principalmente quando as opções de apresentação mais simples do HTML forem ultrapassadas.

A maioria dos browsers utilizados atualmente, apresenta suporte ao XML, com a capacidade de vinculação de dados do DHTML e do CDF, e também trazem suporte ao DOM - Document Object Model (modelo de objetos de documento), ou seja, os elementos utilizados podem ser definidos em XML na forma de objetos, que poderão serem programados através de linguagens script, tornando assim, XML a linguagem ideal para exibir dados dinâmicos em plataformas diferentes.

Além de permitir especificar dinamicamente que os dados se modificam em seus documentos, permitindo também criar fontes de dados que podem ser usadas em uma série de finalidades que, por sua vez, torna possível a interoperabilidade entre aplicativos de um servidor ou em uma máquina cliente e seu site [MOU00].

Segundo [ABI00], XML em sua forma básica, é simplesmente uma sintaxe para transmissão de dados, devendo se tornar o formato universal para troca de dados na Web, apresentando uma especificação que facilita esta troca de dados e a reutilização por múltiplas aplicações, surgindo assim ferramentas para importação e exportação de dados em XML.

8.4 A Aplicação

A tela de abertura/apresentação da aplicação é a mostrada na Figura 36, onde se visualiza um menu geral do lado esquerdo, com as opções de Listagem Completa, Consulta, Visualização Individual, Visualização Individual Completa, o Curso Técnico de Informática, o Curso de Informática, Detalhes, Links para a o Site da Urcamp, Instituto de Informática da UFRGS e para a Página pessoal do autor.

Na parte superior aparece os dados de identificação referentes ao trabalho propriamente dito, e abaixo/direita a tela de apresentação geral (tela inicial).

FIGURA 36. *Tela de Abertura* da Aplicação.

A opção do menu *Listagem Completa*, exibe conforme a Figura 37, a listagem completa de todos os trabalhos cadastrados, a serem manipulados pela aplicação, sendo esta em formato XML.

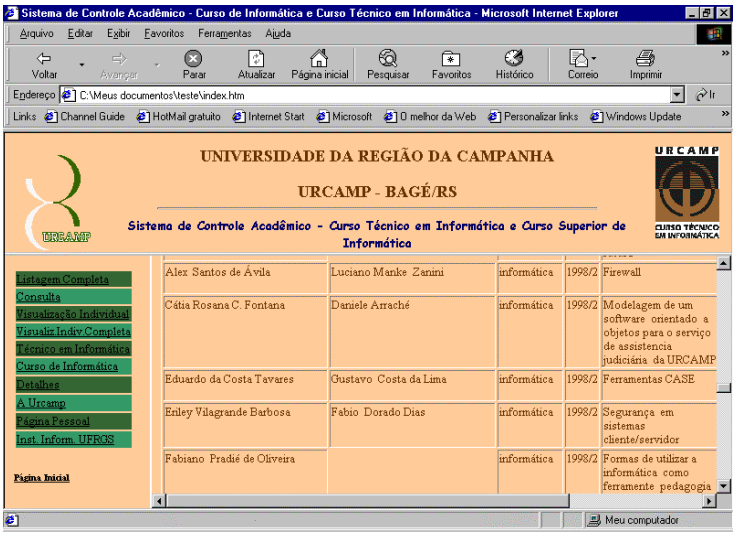

FIGURA 37. Aplicação – Opção *Listagem completa* de todos os trabalhos cadastrados

.A opção *consulta* exibe uma página intermediária onde se escolhe qual o parâmetro da consulta que se deseja, com as opções: consulta por *Curso,* consulta por *Nome do Aluno* , por *Área,* pelo *título* do trabalho, consulta por *professor orientador e* consulta pelo *ano do trabalho,* o que pode ser observado na Figura 38.

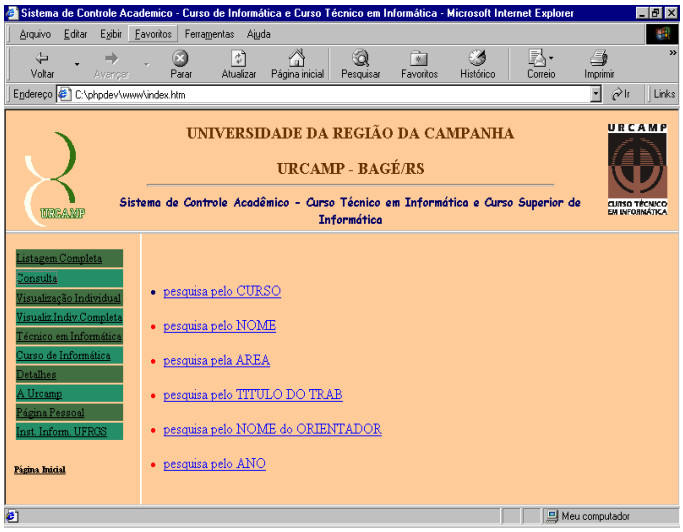

FIGURA 38. Aplicação – Opção Consulta.

O item *Visualização Individual*, Figura 39, exibe a visualização individual completa de cada registro ou trabalho, ou seja, todos os dados do arquivo XML, separando-os por subelementos, ou seja, registro por registro, de modo que através dos botões Avançar e Próximo obtém-se na tela cada um dos trabalhos armazenados no arquivo.

Porém a Visualização Individual Completa, exibe além dos campos já mencionados, um pequeno resumo do trabalho/monografia, Figura 40.

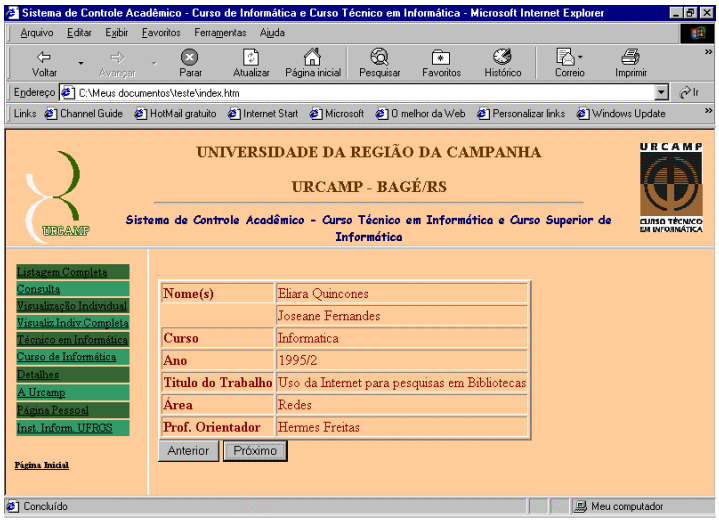

FIGURA 39. Aplicação – Opção *Visualização Individual*.

FIGURA 40. Aplicação – Opção Visualização Individual Completa.

Já os itens, *Curso Técnico de Informática*, e Curso de Informática, descrevem os cursos respectivamente, dando ao visitante noções sobre a estrutura de cada um dos cursos, Figuras 41 e 42, respectivamente.

Após, é exibida a opção *Detalhes*, Figura 43, onde é descrito sucintamente os objetivos da aplicação e a metodologia utilizada..

As opções *Urcamp*, *Página Pessoal e Instituto de Informática da UFRGS*, são links para os referidos Web Sites.

E finalmente a opção *Página Inicial*, retorna a abertura da aplicação.

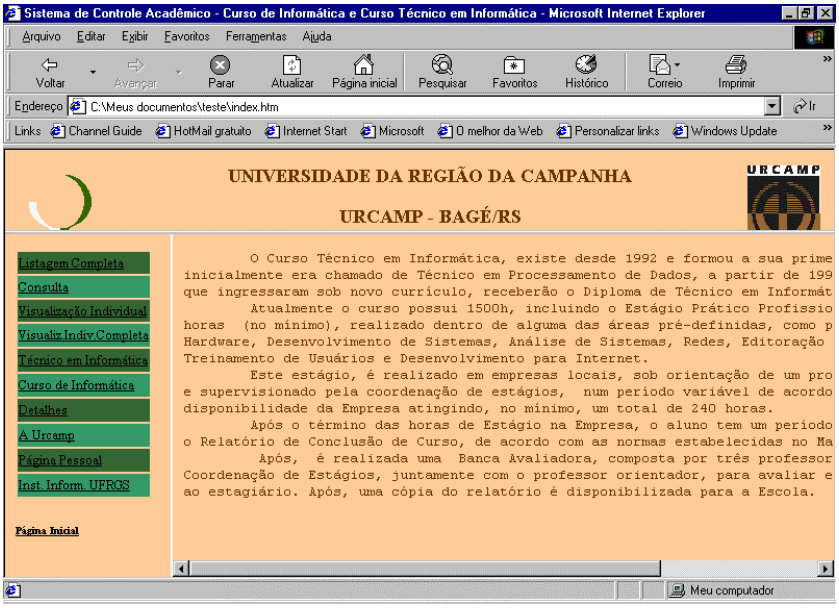

FIGURA 41. Opção *Curso Técnico de Informática*

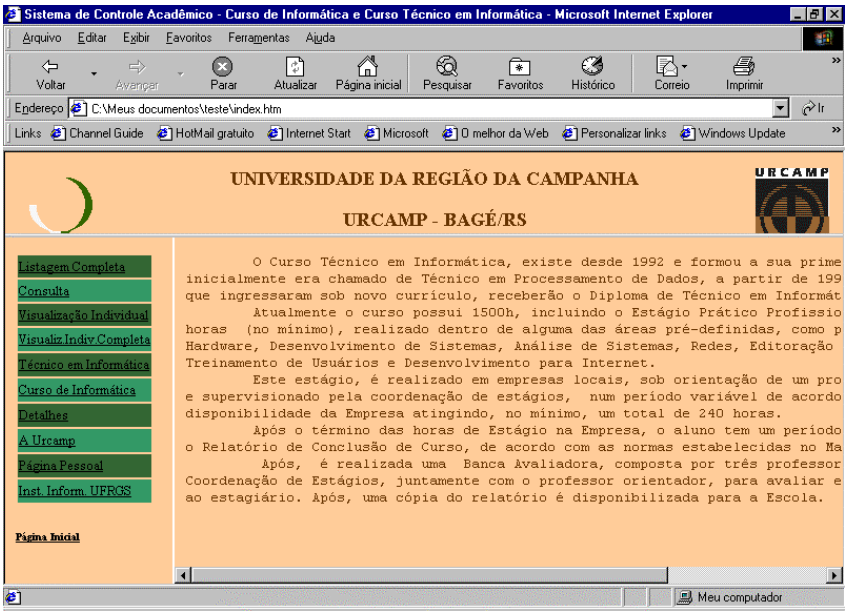

FIGURA 42. Opção *Curso de Informática*.

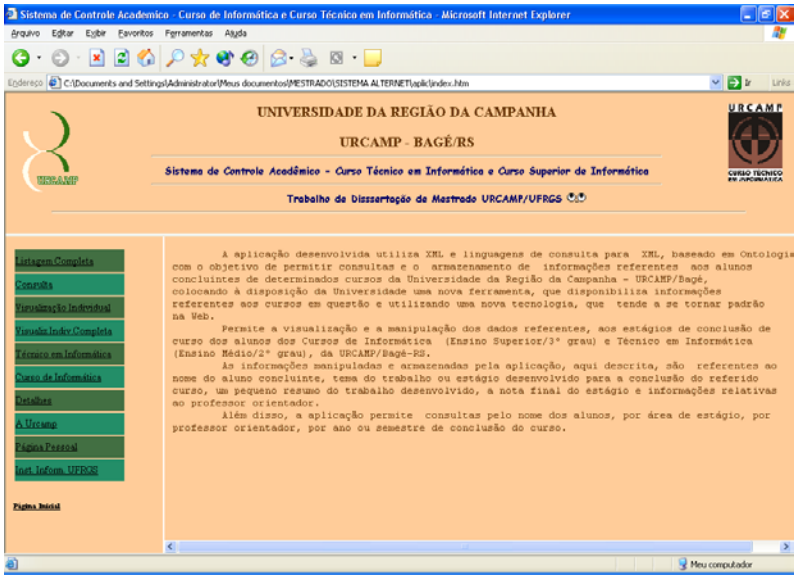

FIGURA 43. Opção *Detalhes* da Aplicação.

8.5 Justificativa da Arquitetura utilizada

Inicialmente foi criada a base de dados em XML, utilizando-se a ferramenta *XML NotePad*, da Microsoft (Figura 44), devido a sua disponibilidade na Web e a praticidade/facilidade no seu uso, cadastrando-se todos os dados referentes aos trabalhos de conclusão de curso, (Figura 45).

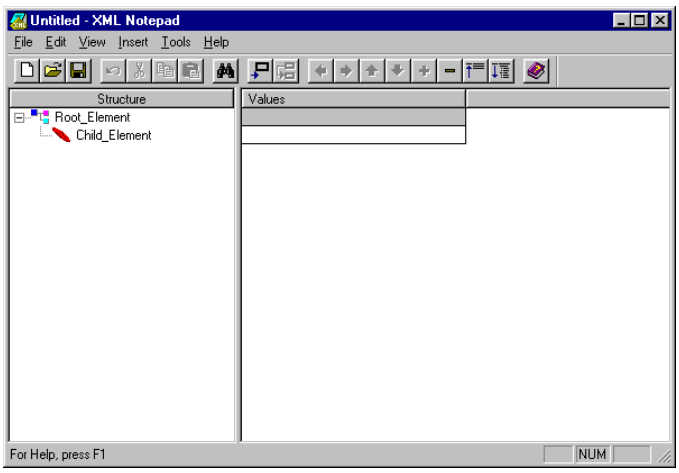

FIGURA 44. *XML Notepad (Microsoft).*

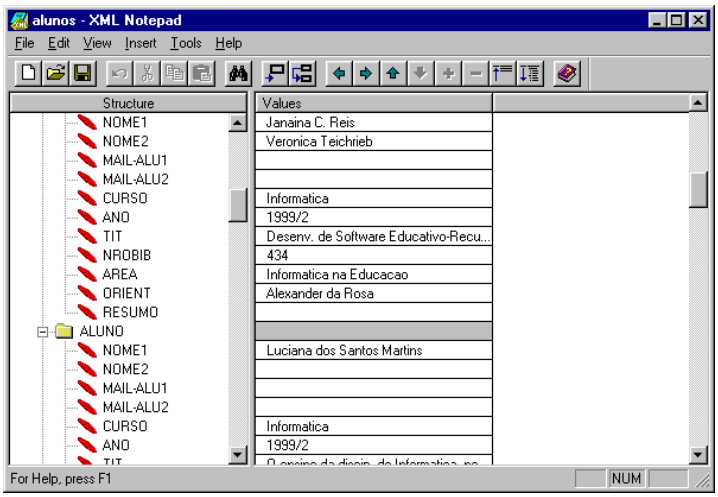

FIGURA 45. Edição do arquivo ***alunos.xml*** (base de dados em XML).

Os dados foram solicitados a coordenação de cada um dos cursos envolvidos no presente estudo, após estes dados foram analisados com relação as informações que seriam relevantes e que realmente deveriam ser disponibilizadas na Web para consulta e/ou pesquisa.

O documento XML foi estruturado da seguinte forma:

- um elemento raiz (*root element*) chamado de ***alunos;***
- subelementos ***nome1*** e ***nome2***, que delimitam os nomes dos alunos autores da monografia, em função de que em um dos cursos aqui citados as monografias poderiam ser desenvolvidas em dupla;
- subelementos ***mailalu-1*** e ***mailalu-2***, que referem-se aos e-mails dos alunos 1 e 2 respectivamente;
- o subelemento ***curso***, refere-se ao curso no qual o aluno concluiu ou desenvolveu a monografia;
- o subelemento ***ano***, refere-se ao ano de conclusão do curso ou desenvolvimento da monografia;
- o subelemento ***tit***, refere-se ao título do trabalho ou monografia desenvolvido;
- o subelemento ***area*** que refere-se a área de estudo da monografia;
- o subelemento ***orient*** que cita o nome do orientador do aluno(s) no trabalho;
- o subelemento ***resumo***, que mostra um pequeno resumo do trabalho, o que não encontra-se preenchido em todos os trabalhos devido a este processo ser um tanto demorado e envolveria uma breve leitura de todos os trabalhos....

A seguir é apresentado um pequeno trecho do arquivo alunos.xml utilizado na aplicação e que evidencia o uso dos subelementos acima descritos.

```
<?xml version="1.0"?>
<ALUNOS>
  <ALUNO>
    <NOME1>Andreia Comim</NOME1>
    <NOME2/>
    <MAIL-ALU1/>
    <MAIL-ALU2/>
    <CURSO>Informatica </CURSO>
    <ANO>1995/2</ANO>
    <TIT>Cereal System - Sistema de Armazenamento de Cereais</TIT>
    <AREA>Analise de Sistemas</AREA>
    <NROBIB/>
    <ORIENT>Fausto</ORIENT>
    <RESUMO>Este trabalho foi desenvolvido para a CESA/Bagé-RS (Companhia Estadual de Silos e
Armazéns). .com o objetivo de controlar  o armazenamento de cereais nos silos da empresa...</RESUMO>
  </ALUNO>
  <ALUNO>
    <NOME1>Eliara Quincones </NOME1>
    <NOME2>Joseane Fernandes</NOME2>
    <MAIL-ALU1/>
    <MAIL-ALU2/>
    <CURSO>Informatica </CURSO>
    <ANO>1995/2</ANO>
    <TIT>Uso da Internet para pesquisas em Bibliotecas</TIT>
    <NROBIB>441</NROBIB>
    <AREA>Redes</AREA>
    <ORIENT>Hermes Freitas</ORIENT>
    <RESUMO/>
  </ALUNO>
<ALUNO>
    <NOME1>Fabio Silveira </NOME1>
    <NOME2>Rodrigo Cardoso</NOME2>
    <MAIL-ALU1/>
    <MAIL-ALU2/>
    <CURSO>Informatica</CURSO>
    <ANO>1995/2</ANO>
    <TIT>Multimidia aplicada ao Turismo</TIT>
    <NROBIB>443</NROBIB>
    <AREA>Sistemas de Informacao</AREA>
    <ORIENT>Ricardo Bernardes</ORIENT>
    <RESUMO/>
 </ALUNO>
```

Para o desenvolvimento da tela inicial da aplicação foi usado a linguagem HTML e recursos de frames para criação das três partes da janela. A parte superior da tela, onde consta os dados gerais de identificação do sistema é fixa, o conteúdo que altera é apenas a parte do frame da direita (conteúdo) a medida que escolhe-se as diferentes opções do menu do frame da esquerda (menu).

O menu foi montado também utilizando-se HTML e, através do comando <table> </table> que através de uma tabela cria o layout usado. Em cada uma das opções utilizou-se links para arquivos e Websites de acordo com a opção escolhida.

A aplicação está disponível por http nas seguintes URLs:
www.attila.urcamp.tche.br/carla/index.htm e **www.alternet.com.br/carla/aplic/index.htm**

Na primeira opção do menu, Listagem completa, o link é para o arquivo *tela1.htm*, Figura 46, abaixo. Inicialmente é montada uma tabela onde os cabeçalhos correspondem a todos os elementos descritos para cada trabalho e/ou monografia do arquivo *alunos.xml*, que neste caso é a base de dados. Após são exibidos todos os dados de todos os elementos do arquivo, listando assim todos os trabalhos cadastrados até a data atual.

```
<font size="1"></font>
<body text="#663300" bgcolor="#FFCC99" link="#800000" vlink="#800000">
<xml id=xmldso async="false" src="alunos.xml">
</xml>
 <font size="1">
<b>
<TABLE border=1 dataSrc=#xmldso id=table>
  <THEAD>
   <TR>
    <TH>
    <small><TD COLSPAN="2" ALIGN="LEFT"><b>Nome(s)  Aluno(s)</b></TD></small> </b>
    <small><TH>Curso</small>
    <small><TH>Ano </small>
    <small><TH>Título </small>
    <small><TH>Área </small>
    <small><TH>Orientador </small>
    </TR>
 </THEAD>
</b>
  <TBODY>
 <TR>
    <TD ColSpan="2" vAlign="top">
      <small><DIV noWrap dataFld=NOME1></DIV></small>
    <TD noWrap rowSpan=1 vAlign=top>
      <small><DIV dataFld=NOME2></DIV></TD></small>
    <TD vAlign=top>
      <small><DIV dataFld=CURSO></DIV></TD></small>
    <TD vAlign=top>
      <small><DIV dataFld=ANO></DIV></TD></small>
    <TD vAlign=top>
      <small><DIV dataFld=TIT></DIV></TD></small>
    <TD vAlign=top>
      <small><DIV dataFld=AREA></DIV></TD></small>
    <TD vAlign=top>
      <small><DIV dataFld=ORIENT></DIV></small>
</TD>
</TR>
</font>
</TBODY>
</TABLE>
</BODY>
</HTML>
```

FIGURA 46. Arquivo *tela1.htm*

A segunda opção do menu, *consulta*, quando ativada, exibe uma segunda janela onde se escolhe o item pelo qual se deseja consultar, por curso, por nome, por área de estudo, pelo título do trabalho, pelo nome do orientador e pelo ano, Figura 40.

Após a escolha são chamados os respectivos arquivos que tratam cada uma dessas opções, normalmente solicitando ao usuário a informação na qual será feita a busca no arquivo, por exemplo o nome do aluno, o ano de conclusão do curso, conforme mostra a Figura 47.

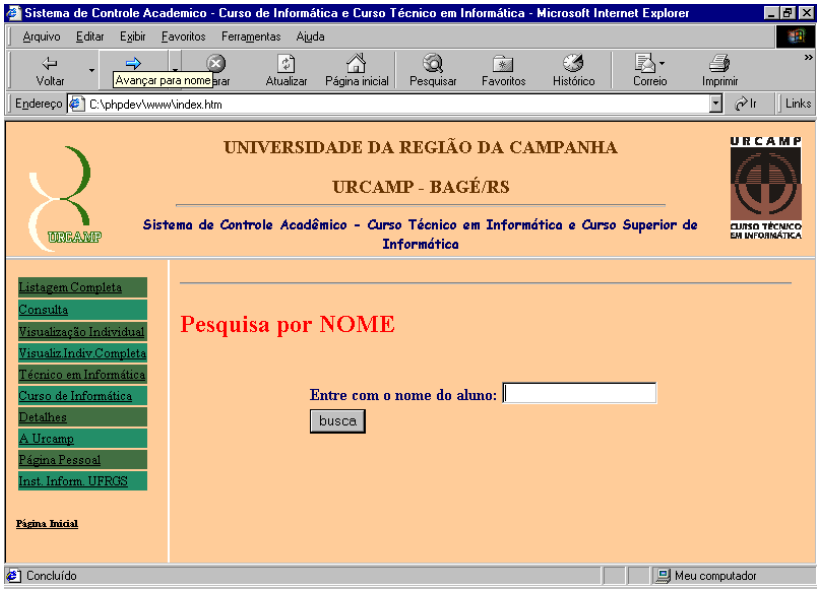

FIGURA 47. Consulta pelo nome do aluno. (arquivo *nome.htm*)

Como as opções de consulta são praticamente as mesmas, vamos exemplificar utilizando a consulta por nome do curso: curso técnico em informática, ou curso superior de informática.

A Figura 48 mostra o arquivo *curso.htm*, onde inicialmente são definidos os cabeçalhos de título e de página, após é criado um formulário com o atributo ACTION definido para a página, *curso.php.. Após existe um elemento SELECT* com opções que permitem ao usuário escolher uma categoria para pesquisa (Curso Técnico ou Curso Superior). A categoria escolhida estará disponível no script *curso.php através da variável $searchBy.* De acordo com o tipo de pesquisa pode haver uma caixa de texto onde a palavra chave para pesquisa pode ser introduzida (variável $searchKeyword) e abaixo um botão de envio.

```
<HTML>
   <HEAD>
    <font color="#ff0000">
       <TITLE>Consulta por NOME do CURSO</TITLE>
   </HEAD>
     <body text="#000066" bgcolor="#FFCC99" link="#3366FF" vlink="#0000FF" alink="#FF0000">
<HR>
        <H2><B>Pesquisa por Curso</B></H2>
  <FORM ACTION="curso.php" METHOD=GET>
         <CENTER>
            <TABLE>
               <TR>
                  <TD>
                      <B>Pesquisa por CURSO</B>
                  </TD>
                  <TD>
                      <SELECT NAME=searchBy  SIZE=1>
                            <OPTION>Informatica
                            <OPTION>Tecnico em Informatica
                      </SELECT>
                <TR>
                   <TD>
                      <B>Palavra Chave</B>
                   </TD>
                   <TD>
                      <INPUT TYPE=TEXT NAME="searchKeyword">
                   </TD>
                 </TR>
                 <TR>
                    <TD> </TD>
                    <TD>
                       <INPUT TYPE="submit" VALUE="Submit">
                    </TD>
                  </TR>

                  </TD>
                  </TR>
             </TABLE>
           </CENTER>
       </FORM>
       <BR><HR>
        </FORM>
   </BODY>
</HTML>
```

FIGURA 48. Arquivo *curso.htm*

O arquivo *curso.php*, Figura 49, após definir os cabeçalhos e títulos iniciais, é gerada uma tabela para apresentar os detalhes dos trabalhos dos alunos.

Dentro da tabela usa-se *require* para o arquivo *gecur.php*, onde é procurado nos dados contidos no arquivo XML os trabalhos de acordo com o critério de busca solicitado (no exemplo: nome do curso).

O resultado é montado e mostrado em uma tabela que chama a função readBookInfo( ) , de acordo com o conteúdo da variável $SearchKeyword.

```
<HTML>
 <HEAD>
   <TITLE>Resultados da Pesquisa</TITLE>
 </HEAD>
<body text="#000066" bgcolor="#FFCC99" link="#3366FF" vlink="#0000FF"
alink="#FF0000">
    <B>Resultados da Pesquisa</B>
       <HR><BR>
<TABLE BORDER=1>
<THEAD>
  <TR>
      <TH> NOME1 </TH>
      <TH> NOME2 </TH>
      <TH> CURSO </TH>
      <TH> ANO </TH>
      <TH> TIT </TH>
      <TH> NROBIB </TH>
      <TH> AREA </TH>
      <TH> ORIENT </TH>
      <TH> RESUMO </TH>
   </TR>
</THEAD>
<TBODY font bgcolor="#a68064">

<?php
    require("gecur.php");
    $books = readBookInfo();
       if (strcmp($searchBy, "CURSO") == 0)   {
       if (($book = searchBookByISBN($book, $searchKeyword)))
         printBookInfo($book["NOME1"], $book["NOME2"],
         $book["CURSO"], $book["ANO"], $book["TIT"],
         $book["NROBIB"], $book["AREA"],
         $book["ORIENT"], $book["RESUMO"]);
            }
elseif (strcmp ($searchBy, "CURSO") == 0)   {
for  ($i=0; $i<count($book); $i++)   {
if (strstr(strtolower(trim($book[$i]["CURSO"])),
strtolower(trim($searchKeyword))))
printBookInfo($books[$i]["CURSO"],
    $books[$i]["NOME1"],
    $books[$i]["NOME2"],
    $books[$i]["CURSO"],
    $books[$i]["ANO"],
    $books[$i]["TIT"],
    $books[$i]["NROBIB"],
    $books[$i]["AREA"],
    $books[$i]["ORIENT"],
    $books[$i]["RESUMO"]);
            }
       }
?>
</TBODY>
</TABLE>
</BODY>
</HTML>
```

FIGURA 49. Arquivo *curso.php*

```
<?php
   $file = "alunos.xml";
   $currentTag = "";
   $NOME1Value = "";
   $NOME2Value = "";
   $CURSOValue = "";
   $ANOValue = "";
   $TITValue = "";
   $NROBIBValue = "";
   $AREAValue = "";
   $ORIENTValue = "";
   $RESUMOValue = "";
   $books = array( );
   function startElement ($parser, $name, $attr) {
     global $currentTag, $NOME1Value, $NOME2Value, $CURSOValue, $ANOValue,
     $TITValue, $NROBIBValue, $AREAValue, $ORIENTValue, $RESUMOValue;
     $currentTag = $name;
    function endElement ($parser, $name)  {
     global $NOME1Value, $NOME2Value, $CURSOValue, $ANOValue, $TITValue, $NROBIBValue,
         $AREAValue, $ORIENTValue, $RESUMOValue;
     if (strcmp ($name, "ALUNO") == 0)  {
           $books[] = array ("NOME1" => $NOME1Value,
                          "NOME2" => $NOME2Value,
                         "CURSO" => $CURSOValue,
                           "ANO" => $ANOValue,
                           "TIT" => $TITValue,
                        "NROBIB" => $NROBIBValue,
                          "AREA" => $AREAValue,
                        "ORIENT" => $ORIENTValue,
                        "RESUMO" => $RESUMOValue);
   $NOME1Value = "";
   $NOME2Value = "";
   $CURSOValue = "";
   $ANOValue = "";
   $TITValue = "";
   $NROBIBValue = "";
   $AREAValue = "";
   $ORIENTValue = "";
   $RESUMOValue = "";

#   $authorCount = 0;
#   $descriptionValue = "";
         }
     }
function characterData($parser, $data)  {
     global $NOME1Value, $NOME2Value, $CURSOValue, $ANOValue, $ANOValue,
          $TITValue, $NROBIBValue, $AREAValue, $ORIENTValue, $RESUMOValue;
         if (strcmp ($currentTag, "NOME1") == 0)  {
           $NOME1Value .= $data;
         } elseif (strcmp ($currentTag, "NOME2") == 0)  {
           $NOME2Value .= $data;
         } elseif (strcmp ($currentTag, "CURSO") == 0)  {
           $CURSOValue .= $data;
         } elseif (strcmp ($currentTag, "ANO") == 0)  {
           $ANOValue .= $data;
         } elseif (strcmp ($currentTag, "TIT") == 0)  {
           $TITValue .= $data;
         } elseif (strcmp ($currentTag, "NROBIB") == 0)  {
           $NROBIBValue .= $data;
         } elseif (strcmp ($currentTag, "AREA") == 0)  {
           $AREAValue .= $data;
         } elseif (strcmp ($currentTag, "ORIENT") == 0)  {
           $ORIENTValue .= $data;
         } elseif (strcmp ($currentTag, "RESUMO") == 0)  {
           $RESUMOValue .= $data;
         }
        }
 function externalEntityHandler ($parser, $entityName, $base,
                                 $systemId, $publicId)  {
         global $descriptionValue;
         if (!systemId)
           return false;
         $descriptionValue = $systemId;
           return true;
```

FIGURA 50. Arquivo *gecur.php*

```
function externalEntityHandler ($parser, $entityName, $base,
                                $systemId, $publicId)  {
          global $descriptionValue;
          if (!systemId)
            return false;
          $descriptionValue = $systemId;
            return true;
       }
function readBookInfo()  {
   global $file, $books;
   $xml_parser = xml_parser_create();
   xml_set_element_handler($xml_parser, "startElement", "endElement");
   xml_set_character_data_handler($xml_parser, "characterData");
  xml_set_external_entity_ref_handler($xml_parser, "externalEntityHandler");
   xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, false);
   if (!($fp = fopen($file, "r")))  {
   die("Could not open $file for reading");
         }
   while (($data = fread($fp, 4096)))  {
   if (!xml_parse($xml_parser, $data, feof($fp))) {
   die(sprintf("XML error at line %d column %d",
   xml_get_current_line_number($xml_parser),
   xml_get_current_column_number($xml_parser)));
     }
    }
   xml_parser_free($xml_parser);
   return $book;
      }

function printBookInfo($NOME1Value, $NOME2Value, $CURSOValue, $ANOValue,
                  $TITValue, $NROBIBValue,$AREAValue, $ORIENTValue,
                  $RESUMOValuE) {

            print "<TR>";
            print "<TD>$NOME1Value</A></TD>";
            print "<TD>";

#            print " $NomeValue[$j] ";
            print " $NOME1Value ";
            }
            print "</TD>";
            print "<TD>$NOME2Value</TD>";
            print "<TD>$CURSOValue</TD>";
            print "<TD>$ANOValue</TD>";
            print "<TD>$TITValue</TD>";
            print "<TD>$NROBIBValue</TD>";
            print "<TD>$AREAValue</TD>";
            print "<TD>$ORIENTValue</TD>";
            print "<TD>$RESUMOValue</TD>";
            print "</TR>";
        }
function searchBookByISBN($books, $Nome)  {
   for ($i = 0; $i < count($books); $i++)  {
   if (strcmp(trim($books[$i]["Nome"]), trim($Nome)) == 0)  {
       return $books[$i];
        }
    }
  return null;
}
?>
```

FIGURA 51. Arquivo *gecur.php* (continuação da Figura 50).

9 Conclusões

Atualmente existe uma grande quantidade de informações disponíveis na Web, de uma forma não estruturada ou não organizada, o que dificulta a busca da informação desejada de forma rápida e correta.

Com o surgimento e o propósito da XML, as informações seriam organizadas de uma forma estruturada, de modo que poderiam ser pesquisadas ou acessadas pelas mais diferentes linguagens e ferramentas de consulta, proporcionando assim buscas mais eficientes.

Atualmente, observa-se a tendência da XML em se tornar a linguagem padrão na Web, sendo que a representação das informações pode ser dirigida para este novo padrão, pois disponibiliza um bom intercâmbio de informações e a produção de documentos eletrônicos.

Neste sentido, o trabalho buscou inicialmente, um estudo aprofundado do que existia na época, disponível sobre a linguagem XML, a sintaxe da linguagem, linguagens de consulta disponíveis para XML, e a extração destes dados baseado em Ontologias.

Após, foi construída uma aplicação visando aplicar os conceitos adquiridos e estudados, utilizando-se de dados fornecidos e disponibilizados pela Universidade.

Com isso, foi permitido o estudo de um novo padrão que vem alcançando um certo destaque na World Wide Web, além do conhecimento de ferramentas, linguagens e novos conceitos, o que pode ser verificado através da aplicação propriamente dita, disponibilizada na URL citada.

Em breve a maioria dos dados disponibilizados na Web estarão no formato XML, pois atualmente, existe um grande número de profissionais dedicando-se a conversão de sua base de dados para XML, e observa-se que desenvolvedores de ferramentas genéricas para a Web vêm apoiando e estudando esta nova tecnologia.

Com isso, será facilitada a troca de informações na Web abrindo novos focos de estudo que envolvam uma busca mais eficiente e rápida.

A nível pessoal, obteve-se um amplo conhecimento na área de desenvolvimento para a Web, devido a revisão bibliográfica e o estudo do que estava sendo destacado nesta área no momento, permitiu também o desenvolvimento de uma ferramenta/aplicação utilizando essas novas tecnologias, que foi disponibilizada a URCAMP-Bagé/RS, para uso na sua Home-Page.

Também salienta-se que no decorrer deste trabalho, houve uma mudança de domicílio, e o desenvolvimento do presente trabalho engrandeceu o curriculum de maneira que foi possível o ingresso em duas Universidades Catarinenses.

O trabalho poderá ser aperfeiçoado e adequando a medida com que estas novas ferramentas e tecnologias forem disponibilizadas, otimizando assim as consultas, a exibição de um mesmo conteúdo em diferentes dispositivos, assim como uma busca mais rápida e eficiente das informações na WWW.

A aplicação desenvolvida neste trabalho, também poderá ser aplicada ao meio acadêmico no qual estamos inseridos atualmente, duas Universidades particulares do Estado de SC, mais precisamente em Chapecó. Existe a possibilidade bastante palpável deste estudo ser alvo de pesquisa em uma Universidade Local, uma vez que existem grupo de estudo interessados em dar prosseguimento ao trabalho aqui apresentado.

BIBLIOGRAFIA

[ABI 99] ABITEBOUL, S. et al**. The Lorel Query Language for Semistructured Data**. Cidade: Universidade de Stanford, 1999. Disponível em:<http://www-db.stanford.edu> Acesso em 20 fev. 2000.

[ABI 2000] ABITEBOUL, S.; BUNEMAN, P.; SUCIU, D. **Gerenciando Dados na Web**. Tradução de Mônica Cardia. Rio de Janeiro: Campus, 2000.

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

[ARO 97] AROCENA, G. O.; MENDELSON, A. O.; MIHAILA,G. A. **Aplicattions of a Web Query Language**. In: INTERNATIONAL WORLD WIDE WEB CONFERENCE, WWW, 6., 1997, Santa Clara, US. Proceedings … [Stanford University], 1997.

[ARO 98] AROCENA,G. O.; MENDELSON, A. O.; MIHAILA,G. A. **Query Languages for the Web.** 1998**.** Disponível em: <http://www.w3.org/TandS/QL/QL98/pp/wql.html>. Acesso em: 30 out. 1999.

[BAR98] BARU, C.; et al. **Features and requirement for na XML view definition language lessons form XML information mediation**. In: W3C Workshop on Query Languages 1998, Boston. Disponível em: <http://www.w3.org/TandS/QL/QL98/pp/xmas.html. Acesso em: 20 ago.2000.

[CAS 2001] CASTAGNETTO, J. et al. **Professional PHP:** programando. São Paulo, Makron Books. 2001.

[CER 98] CERI, Stefano et al. **XML-GL:** a graphical Language for Querying and Reshaping XML Documents**.** Milano: Universidade de Milano, 1998. Disponível em: <http://www.w3.org/TandS/QL/QL98/pp/xml-gl.html>. Acesso em: 30 out. 1999.

[DEU 99] DEUTSCH, A. et al. **A query language for XML.** In.: International World Wide Web Conference, WWW, 8., 1999, Toronto. Disponível em: <http://www.research.att.com/~mff/xmlql/doc/files/final.html.> Acesso em: 03 fev. 2000.

[DOR99] DORNELES, C. F., HEUSER, C. A.; SILVA JUNIOR, A. S. **Extração de Informação em Documentos XML baseada em Ontologias**. In: WORKSHOP DO PROJETO IDOC, 1999. Gestão de Conhecimento Baseado em Documentos: anais. [Paraná: CEFET], 1999.

[ERD 2000] ERDMAN, M.; STUDER, R. Ontolgies as Conceptual Models for XML Documents. AIFB. Germany: Universituy of Karlsruhe. Disponível em: <http://www.ontoserver.aifb.unikarlsruhe.de/dtdmaker.html> Acesso em: 20 out. 2000.

[GRA 99] MCGRATH, Sean. **XML:** aplicações Práticas. Tradução de Vitor Hugo de Paixão Alves. Rio de Janeiro: Campus.1999.

[KAD99] KADE, A.; HEUSER,C.A. **Tendências em Linguagens de Consulta para documentos XML.** In: WORKSHOP DO PROJETO IDOC, 1999. Gestão de Conhecimento Baseado em Documentos: anais. [Paraná: CEFET], 1999.

[LIG 99] LIGHT, Richard. **Iniciando em XML**. Tradução de Neilande de Moraes. São Paulo: Makron Books, 1999.

[MAI98] MAIER, D**. Database desiderata for na XML query language**. In: WORKSHOP ON QUERY LANGUAGES, 1998, Boston,US.. Disponível em: <http://www.w3.org/TandS/QL/QL98/pp/maier.html> Acesso em: 15 ago. 2000.

[MAR 2000] MARCHAL, Benoit. **XML:** Conceitos e Aplicações. São Paulo: . Bekerley, 2000.

[MEL 2000] MELLO, R. S. **Aplicação de Ontologias à Banco de Dados Semi-Estruturados**. 2000. (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[MOU 2000] MOULTIS, Natanya Pitts; KIRK, Cheryl. **XML Black Book**. São Paulo, Makron Books, 2000. Traduzida por Ariovaldo Griessi.

[PFE 98] PFEIFER, Ralf I. **XML Tutorial for Programmers**. IBM XML Tecnology Group, 1998. Disponível em <http://www. review.software.ibm.com/developer/library/tutorialprog/overview.html> Acesso em: 06 jul. 1999.

[ROB 99] ROBIE, Jonathan. **The design of XQL**. Texel Reseach. 26/03/1999. Disponível em: <http://www.texcel.no/whitepapers/xql-desingn.html> Acesso em: 27 fev. 2000.

[ROS 98] DEROSE, Steven J. **Xquery**: a unified syntax for linking and querying general XML documents. 1998. Disponível em: <http://www.w3.org/TandS/QL/QL98/pp/xquery.html> Acesso em: 30 out. 1999.

[SIL99] SILVA JUNIOR, A. S.; HEUSER,C. A.; DORNELES, C.F**.** **Materialização de Visões Semi-Estruturadas em Banco de Dados Relacional**. IN: WORKSHOP DO PROJETO IDOC. 1999. Gestão de Conhecimento Baseado em Documentos: anais. [Paraná: CEFET], 1999.

[VIS 98] VISSILAKIS, C. **Object Query Language**. University of Athens,. 27/05/1998. Disponível em <http://www. .di.uoa.gr/~toobis// seminar/OQL/tsld001.htm> Acesso em: 10 mar. 2000

[W3C 99] W3C. **Extensible Markup Language (XML) 1.0**. Disponível em <http://www.w3.org/TR/1999/REC-xml-19980210. Acesso em: 30 out. 1999.

[W3C 00a] W3C. **XML Schema Part 1:** structures.working draft. Disponível em <http://www.w3.org/TR/xmlschema-1/> Acesso em: 01 mar. 2000.