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

Consultando fontes de dados XML heterogêneas através de modelos conceituais

por

SANDRO DANIEL CAMILLO

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

Prof. Dr. Carlos Alberto Heuser
Orientador

Porto Alegre, abril de 2003.

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Camillo, Sandro Daniel

Consultando fontes de dados XML heterogêneas através de modelos conceituais / por Sandro Daniel Camillo. — Porto Alegre: PPGC da UFRGS, 2003.

66 f.: il.

Dissertação (mestrado) — Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2003. Orientador: Heuser, Carlos Alberto.

1. XPath. 2. XQuery. 3. CXPath. 4. CXQuery. 5. Fontes XML heterogêneas. 6. Tradução de consultas. 7. Modelo Conceitual Global. I. Heuser, Carlos Alberto. II. Título.

Sumário

Lista de Abreviaturas

CXQuery	Conceptual XQuery
CXPath	Conceptual XPath
XML	eXtensible Markup Language
DTD	Document Type Definition
W3C	World Wide Web Consortium
ODMG	Object Data Management Group
ORM	Object with Role Model

Lista de Figuras

Lista de Tabelas

Resumo

XML é um padrão da W3C largamente utilizado por vários tipos de aplicações para representação de informação semi-estruturada e troca de dados pela Internet. Com o crescimento do uso de XML e do intercâmbio de informações pela Internet é muito provável que determinadas aplicações necessitem buscar uma mesma informação sobre várias fontes XML relativas a um mesmo domínio de problema.

No intuito de representar a informação dessas várias fontes XML, o programador é obrigado a escolher entre muitas estruturas hierárquicas possíveis na criação dos esquemas de seus documentos XML. Um mesmo domínio de informação, desta maneira, pode então ser representado de diferentes formas nas diversas fontes de dados XML existentes. Por outro lado, verifica-se que as linguagens de consulta existentes são fortemente baseadas no conhecimento da estrutura de navegação do documento. Assim, ao consultar uma determinada informação semanticamente equivalente em várias fontes é necessário conhecer todos os esquemas envolvidos e construir consultas individuais para cada uma dessas fontes.

Em um ambiente de integração, entretanto, é possível gerar um modelo global que representa essas fontes. Ao construir consultas sobre um modelo global, temos consultas integradas sobre estas fontes. Para se atingir esse objetivo, contudo, devem ser resolvidos os problemas da heterogeneidade de representação dos esquemas XML.

Dessa forma, com base em uma abordagem de integração de esquemas, o presente trabalho define a linguagem de consulta CXQuery (Conceptual XQuery) que possibilita a construção de consultas sobre um modelo conceitual.

Para possibilitar o retorno dos dados das fontes nas consultas feitas em CXQuery, foi desenvolvido um mecanismo de tradução da linguagem CXQuery para a linguagem de consulta XQuery 1.0. A linguagem XQuery 1.0 é umas das linguagens mais utilizadas para o acesso as fontes XML e permite que os dados possam ser retornados ao usuário. Para possibilitar essa tradução, foi definida uma metodologia de representação da informação de mapeamento através de visões XPath. Essa metodologia é relativamente eficaz no mapeamento das diferentes representações das fontes XML.

Palavras-chave: XPath, XQuery, CXPath, CXQuery, Fontes XML heterogêneas, Tradução de consultas, Modelo Conceitual Global.

TITLE: "QUERYING HETEROGENEOUS XML DATA SOURCES THROUGH A CONCEPTUAL SCHEMA"

Abstract

XML is a W3C standard widely used by some types of applications for representing semi-structured information and exchanging data over the Internet. With the increase on use of XML and information exchange over the Internet, many times certain applications need to search the same semantic information on XML sources of the same domain of problem.

In order to represent the information of these XML sources, the programmer need to choose between many possible hierarchical structures during the schema creation process. The same information domain, in this way, can then be represented in different ways in the several XML data sources. On the other hand, it is known that the existing query languages are strongly based on the knowledge of the document navigational structure. As a consequence, building queries to access the same semantic information on different XML sources requires knowledge of all the schemas involved and the formulation of individual queries for each one of these data structures.

In an integration environment, however, it is possible to generate a global model that represents these XML sources. If a query is constructed over a global model, we have integrated queries over these sources. Nevertheless, to reach this objective, the problems of heterogeneous XML schema representation must be solved.

Therefore, over an integration environment approach, this work defines the CXQuery (Conceptual XQuery) query language. CXQuery makes it possible to build queries over a conceptual model.

To be able to obtain XML results from queries in CXQuery, we developed a translation mechanism that translates a query in CXQuery to a query in XQuery 1.0. Xquery 1.0 is one of the most used languages to access XML data sources, and is also the W3C XML query standard. The translation mechanism is based on mapping information provided by XPath views, which are constructed during the integration process. We claim that this is an efficient approach to deal with different XML data source representation.

Keywords: XPath,XQuery,CXPath,CXQuery,Heterogeneous XML Sources,Query Translation,Global Conceptual Model.

1 Introdução

XML é um padrão da W3C [W3C 2001] largamente utilizado por vários tipos de aplicações para representação de informação semi-estruturada e troca de dados pela Internet. Diversos domínios de aplicação, como *referências bibliográficas* [DBL 2002, SIG 99] e *comércio eletrônico*, representam informação em XML.

Com o crescimento do uso de XML e do intercâmbio de informações pela Internet, tornou-se necessária a criação de mecanismos para possibilitar a integração de dados de fontes XML heterogêneas, uma vez que é muito provável que determinadas aplicações necessitem buscar uma mesma informação sobre várias fontes XML relativas a um mesmo domínio de problema.

Ao representar informações em uma fonte XML, o projetista da fonte deve escolher entre as muitas estruturas hierárquicas que podem representar a informação desejada em XML. Uma mesma informação conceitual, desta maneira, pode então ser representada com diferentes estruturas nas diversas fontes de dados XML existentes.

Se uma aplicação necessita realizar uma consulta integrada sobre estas fontes, deve resolver os problemas da heterogeneidade de representação dos esquemas XML.

O fato de existirem diferentes representações de uma mesma informação vem criando grandes e clássicos desafios na área de integração de sistemas em geral, desde os primórdios da informática [OSZ 99].

No âmbito de XML, verifica-se que as linguagens de consulta existentes são fortemente baseadas no conhecimento da estrutura de navegação do documento. Assim, um dos problemas verificados é de que ao consultar uma informação semanticamente equivalente em várias fontes, será necessário conhecer todos os esquemas envolvidos e construir consultas individuais para cada uma dessas fontes.

Dessa forma pode-se imaginar um cenário onde o usuário seja obrigado a estudar inúmeros esquemas para construir outras inúmeras consultas diferentes para cada fonte XML, normalmente tornando a operação onerosa ou inviável.

Diante desse problema, os objetivos principais desta dissertação são os seguintes:

1. propor uma abordagem que permita a construção de consultas sobre um modelo conceitual global que represente um conjunto de fontes XML,

2. promover a tradução desta consulta global para as consultas de cada uma dessas fontes levando em conta seus esquemas específicos.

Este trabalho está inserido dentro da arquitetura de integração BInXS [MEL 2000] que propõe uma arquitetura de integração de fontes XML através de um modelo conceitual. Essa arquitetura trata do problema da integração dos esquemas e da criação de um modelo conceitual. Este modelo conceitual define um esquema global e abstrato de várias fontes XML heterogêneas. Os resultados da presente dissertação, entretanto, não são específicos do ambiente BInXS, podendo ser aplicados a outros ambientes similares.

Dessa forma, com base nesta abordagem de integração e de um modelo conceitual definido, o presente trabalho buscará definir:

- uma linguagem de consulta para este modelo,
- a metodologia de representação da informação de mapeamento
- e um mecanismo de tradução da linguagem sobre o modelo conceitual global para uma linguagem de consulta que acesse as fontes XML.

O escopo definido para este trabalho foi concentrado na tradução das consultas, o problema da decomposição de consultas não faz parte do escopo do presente trabalho. As consultas são traduzidas para uma fonte qualquer se a mesma possuir informação que possa ser traduzida de maneira independente das outras fontes.

Esta dissertação está organizada como segue. O capítulo 2 apresenta o modelo conceitual global utilizado, as linguagens de consulta relacionadas e outras abordagens de integração pesquisadas. O capítulo 3 apresenta CXQuery(Conceptual XQuery), a linguagem de consulta definida neste trabalho. O capítulo 4 apresenta a metodologia de representação da informação de mapeamento e o mecanismo de tradução das consultas do modelo global para as fontes XML. O capítulo 5 é dedicado às conclusões. Em anexo são apresentadas a gramática da linguagem CXQuery, uma breve descrição do protótipo desenvolvido e os diagramas dos autômatos da análise léxica.

2 Linguagens de Consulta e Ambientes de Integração

Neste capítulo é apresentado um modelo conceitual global dentro da arquitetura de integração BInXS [MEL 2000] que permite a representação abstrata e integrada das fontes XML. A seguir, são apresentadas linguagens de consulta existentes que podem ser usadas para a construção de consultas sobre o modelo conceitual. Por fim, são apresentados os trabalhos que definem arquiteturas relacionadas ao escopo desta dissertação.

2.1 Modelo conceitual global para consulta à fontes XML heterogêneas

Alguns modelos lógicos, como os esquemas XML, devem escolher uma forma hierárquica (um-para-vários) para representar relações vários-para-vários existentes no mundo real. Dessa forma, as fontes XML heterogêneas podem conter relacionamentos equivalentes em diferentes sentidos de navegação.

Estes problemas são inerentes à estrutura hierárquica que um esquema XML possui e já foram enfrentados no passado na área de Bancos de Dados Hierárquicos [DAT 86]. Algumas adaptações como inserção de relacionamentos lógicos bi-direcionais em parelhas virtuais foram propostas [DAT 86] na tentativa de representar relações vários-para-vários em esquemas hierárquicos. Estas soluções poderiam ser aplicadas em XML com o uso de chaves de referência de maneira similar a [VID 2002], entretanto, não são soluções concisas e forçam um complexo gerenciamento das estruturas envolvidas.

Muitas vezes é necessário representar em XML relações vários-para-vários do mundo real. É o caso de um domínio de publicações, onde artigos podem ser escritos por vários autores e autores podem escrever vários artigos.

Um modelo global que represente um conjunto de fontes XML relativo a esse domínio deve ter a capacidade de representar as relações vários-para-vários existentes do mundo real.

Esse é um dos motivos pelos quais, diferentemente de trabalhos que propõem modelos globais em XML como [MAN 2001], optou-se por um modelo conceitual global não hierárquico. No modelo conceitual, a representação da navegação pelos relacionamentos é feita de maneira abstrata permitindo relacionamentos vários-para-vários e navegação sem sentido obrigatório.

O modelo global que possibilitará a construção de consultas sobre várias fontes XML, portanto, é um modelo a nível conceitual. Este modelo é uma abstração de alto nível para dados de um domínio de aplicação, uma vez que representa um conjunto de entidades do mundo real e seus relacionamentos. Comparados com modelos de dados lógicos, modelos conceituais apresentam um maior poder de expressão, pois não estão vinculados a nenhuma estrutura de dados (tabela, árvore, entre outras) particular [BAT 92].

Esta característica é importante uma vez que o modelo global se propõe a ser uma representação global que consiga integrar vários esquemas de fontes XML heterogêneas.

O nível de abstração de um modelo conceitual permite que diversos esquemas XML possam ser derivados a partir dele. Nas figuras 2.1 e 2.2 são apresentados representações da estrutura de duas fontes XML. Na figura 2.3 é apresentado o correspondente modelo conceitual.

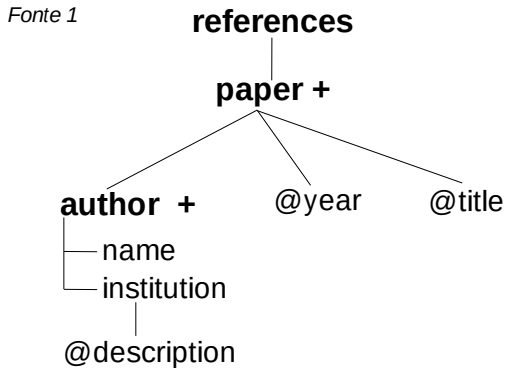


FIGURA 2.1 – Exemplo de fonte de dados XML - Fonte 1

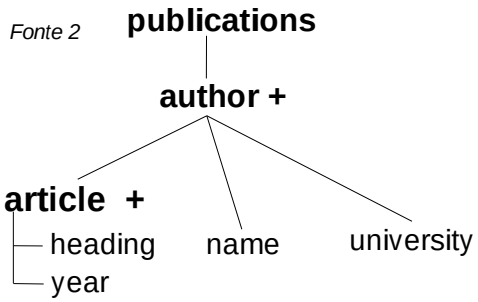


FIGURA 2.2 – Exemplo de fonte de dados XML - Fonte 2

Na fonte 1, mostrada na figura 2.1, o esquema é construído de forma que as publicações são representadas por uma coleção de elementos *paper* e cada *paper* possui todos os seus respectivos autores (elemento *author*) como filhos, ou seja, em um nível hierárquico inferior. O ano de publicação é representado pelo atributo *year* e o título do artigo é representado pelo atributo *title*, ambos do elemento *paper*. O nome do autor é representado pelo elemento *name* e o nome da instituição a que ele pertence está em um atributo *description* do elemento *institution*. Tanto o elemento *name* quanto o elemento *institution*, estão abaixo do elemento *author* na hierarquia.

Já na fonte 2, mostrada na figura 3.2, o esquema é construído de forma que as publicações são representadas por uma coleção de elementos *author* ao invés de *paper*; e, cada *author* possui todos as suas respectivas publicações (desta vez como elemento *article*) como filhos. O título do artigo e o ano desta vez são representados por elementos ao invés de atributos, e com nomes *heading* e *year*. O nome do autor é representado através do elemento *name* e o nome da instituição está diretamente no elemento *university*.

O modelo conceitual que representa as duas fontes apresentadas nas figuras 2.1 e 2.2 pode ser visto na figura 2.3.

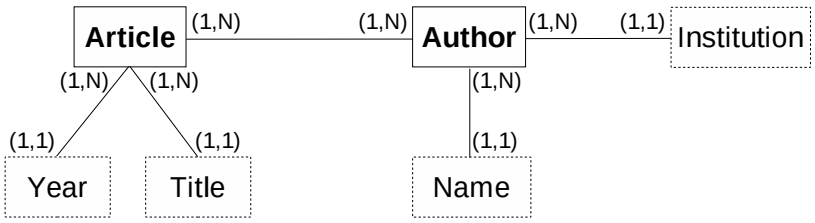


FIGURA 2.3 – Modelo Conceitual

No modelo conceitual da figura 2.3 há os conceitos *year* e *title* que estão relacionados com o conceito *article*, representando o ano e o título do artigo.

Os conceitos *name* e *institution* estão relacionados com o conceito *author* e representam o nome de um autor e a instituição a que a ele pertence.

Conforme a cardinalidade apresentada, estes relacionamentos são do tipo um-para-vários.

Já a navegação existente entre o conceito *author* e o conceito *article* utiliza um relacionamento vários-para-vários. Uma instância do conceito autor pode estar relacionada com várias instâncias do conceito artigo e uma instância do conceito artigo pode estar relacionada com várias instâncias do conceito autor.

O Modelo Conceitual adotado é uma variante do modelo conceitual ORM/NIAM - Object with Role Model/Natural Language Information Analysis Method [HAL 98]. ORM é um modelo baseado em conceitos e relacionamentos semânticos entre conceitos. Dois tipos de conceitos são definidos neste modelo: não-léxicos e léxicos. Um *conceito não-léxico* (retângulo contínuo) modela informação que é composta por outras informações, como por exemplo, *Article*, que possui *Title* e *Year*. Um *conceito léxico* (retângulo tracejado) modela informação que possui um conteúdo diretamente associado, como *Title* e *Year*. Pode haver mais de um relacionamento entre dois conceitos, neste caso, estes relacionamentos devem estar nomeados. Há também a opção de definir papéis para os conceitos em um determinado relacionamento, isto pode ser visto em mais detalhes no capítulo 3. O modelo conceitual também possui restrições de cardinalidade mínima e máxima.

Demais detalhes sobre a integração semântica das fontes e a geração dos modelos conceituais podem ser vistos em [SAN 2001] e [SAN 2001a]. Estes tópicos não fazem parte do escopo desta dissertação.

Dentro dessa abordagem pode ser apresentado o problema de consultas em fontes XML heterogêneas e a utilidade da construção de consultas sobre o modelo conceitual:

Muitas vezes o usuário pode desejar buscar uma determinada informação de várias fontes XML. Como a construção de consultas nas linguagens de consulta XML como XQuery 1.0 e XPath 1.0 e 2.0 [W3C 2001] se baseiam e necessitam do conhecimento navegacional e estrutural do documento XML, o usuário é obrigado a conhecer os detalhes específicos e heterogêneos dos esquemas de cada fonte XML. Conseqüentemente, se for desejo do usuário construir consultas para acessar dados das fontes 1 e 2 apresentadas nas figuras 2.1 e 2.2 dois problemas principais serão verificados:

- o usuário terá que conhecer os dois esquemas por completo.

- o usuário terá de fazer duas consultas diferentes, uma para cada fonte.

Isso pode ser visualizado no exemplo a seguir. Considerando uma consulta para obter os "Títulos dos artigos escritos por Juvenal Antunes"nas fontes 1 e 2 deve-se primeiramente conhecer o esquema XML da fonte 1. A seguir deve-se construir uma consulta XQuery 1.0 para fonte 1, conforme apresentado:

```
for $i in /references/paper
where $i/author/name = "Juvenal Antunes"
return $i/@title
```

Após a construção desta consulta, deve-se conhecer o esquema da fonte 2. Com o conhecimento deste esquema será construída outra consulta para buscar os dados da fonte 2, conforme apresentado a seguir:

```
for $i in /publications/author/article
where $i/../name = "Juvenal Antunes"
return $i/heading
```

Com base no modelo conceitual global, que pode ser visto na figura 2.3, apenas uma consulta será necessária para obter esta informação nas duas fontes. Demais detalhes sobre a construção das consultas sobre o modelo conceitual serão apresentados nos próximos capítulos.

Construindo consultas sobre o modelo conceitual, verifica-se que o custo da operação será menor se comparado ao custo da construção de consultas sobre cada fonte XML heterogênea. Quanto maior for o número de fontes representadas pelo modelo conceitual, maior será a redução nos custos de construção das consultas.

Maiores detalhes sobre as linguagens de consulta que podem ser utilizadas para construção de consultas sobre o modelo conceitual serão apresentadas no próximo capítulo.

2.2 Linguagens de consulta existentes

Para acessar diretamente as fontes de dados XML é utilizada a linguagem XQuery 1.0. A linguagem XQuery 1.0 é uma linguagem padronizada pelo consórcio W3C [W3C 2002] para consulta a fontes de dados XML e troca de dados pela Internet.

As consultas construídas com base no modelo conceitual serão traduzidas para as respectivas consultas XQuery 1.0 em cada fonte XML desejada individualmente.

Um dos objetivos do presente trabalho é definir também a linguagem para que o usuário possa construir consultas sobre o modelo conceitual.

Neste capítulo serão apresentadas as linguagens que podem ser utilizadas com esse propósito, como OQL [ODM 2002], Álgebra para OQL [FEG 2001], SQL [ISO 99], Álgebra ER [CAM 85] e [PAR 84], LOREL [ABI 97], XPath [XML 2001] e [XML 2002] e a própria XQuery [W3C 2002].

Análises dessas linguagens foram efetuadas segundo alguns critérios definidos no decorrer da pesquisa, e as características esperadas para essa linguagem são basicamente:

- **Representação da navegação por relacionamentos através de expressões de caminho.**

  O resultado do processo de tradução será uma linguagem de consulta XML. Como a navegação pelos relacionamentos nas linguagens de consulta XML como XQuery é representada através de expressões de caminho, a tradução fica facilitada se for utilizada uma linguagem com essas características também para consultar sobre o modelo conceitual.

- **Deve ocultar detalhes da estrutura de armazenamento**

  A mesma informação pode ser representada de diferentes formas nas fontes de dados XML. Dessa forma, não faz sentido referenciar detalhes da estrutura de armazenamento, pois a mesma pode ser completamente distinta entre as fontes. A linguagem deve prover uma abstração desses detalhes como o faz o modelo conceitual.

- **Padrão Consolidado**

  Deve ser um padrão conhecido e utilizado pela comunidade a fim de que já tenha sido testado ou validado. O objetivo dessa restrição é evitar padrões incipientes para não ter a tarefa extra de testar a linguagem em si ou consertar possíveis lacunas e inconsistências ainda existentes. Todos esforços devem ser concentrados no foco da dissertação.

- **Operadores Básicos**

  É interessante que a linguagem possua poucos operadores, a exemplo das álgebras. Se a linguagem possuir apenas os operadores básicos, a tarefa de construção do mecanismo de tradução será menos extensa e complexa.

Na próxima seção são apresentados com mais detalhes as linguagens que podem ser utilizadas para construir consultas sobre o modelo conceitual.

2.2.1 Linguagens relacionais - Álgebra ER

Uma opção para a definição da linguagem de consulta para o modelo conceitual é a Álgebra para Modelo Entidade Relacionamento - Álgebra ER [CAM 85] e [PAR 84]. A Álgebra ER possui apenas os operadores básicos de uma linguagem de consulta. Tal característica é interessante para facilitar o desenvolvimento do mecanismo de tradução. Entre as desvantagens, entretanto, pode ser citado o fato de ser uma linguagem pouco conhecida e pouco utilizada pela comunidade.

Exemplo da sintaxe de uma consulta na álgebra ER, onde são solicitados os títulos dos artigos de José da Silva:

π Title (σ name = José da Silva (Author * Article))
π Title (σ name = José da Silva (Article * Author))

Na álgebra ER são utilizadas junções para expressar os relacionamentos. Esta abordagem traz algumas desvantagens para a proposta apresentada. Um dos

problemas é o aumento da complexidade no processo de tradução das consultas pois as junções permitem a comutatividade entre as entidades da junção na construção da consulta, isso impossibilita o mapeamento direto para as fontes XML uma vez que podem ocorrer mapeamentos para caminhos inexistentes na fonte XML. Para solucionar este problema devem ser criados mecanismos adicionais de mapeamento para cada uma das permutações possíveis, com todas as combinações de conceitos e relacionamentos, para o respectivo caminho válido equivalente na fonte XML.

Para expressar todos relacionamentos entre os 3 Conceitos da figura 2.4, por exemplo, podem ser utilizados 8 formas (2^c, onde c é o número de conceitos) quando utiliza-se as junções da Álgebra ER: A*(C*B) ou A*(B*C) ou (B*A)*C ou C*(A*B) ou C*(B*A) ou (A*B)*C ou (B*C)*A ou (C*B)*A

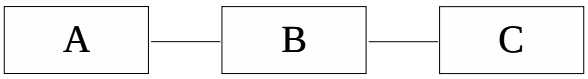


FIGURA 2.4 – Problema da Comutatividade na Álgebra ER

Expressando a navegação pelos relacionamentos através de expressões de caminho apenas 2 formas de uso são permitidas: A.B.C ou C.B.A

O uso de expressões de caminho simplifica a tarefa de tradução das consultas, pois obriga o uso dos relacionamentos de uma forma similar ao utilizado nas principais linguagens de consulta XML, e com um caminho já existente no modelo conceitual. Com isso evita-se que o usuário construa consultas que impeçam um mapeamento direto entre os paradigmas.

2.2.2 Linguagens relacionais - Linguagem SQL

A linguagem de consulta SQL [ISO 99] também pode ser utilizada sobre o modelo conceitual e tem a vantagem de ser amplamente conhecida e testada.

Por outro lado, expressa a navegação pelos relacionamento por junções e possui um conjunto grande de operadores e construções de sintaxe, o que dificulta a construção do mecanismo de tradução proposto.

2.2.3 Linguagens para navegação explícita - Linguagem de consulta OQL

A linguagem OQL é um padrão definido pelo consórcio da ODMG [ODM 2002] para consultas em Banco de Dados orientados a objeto. Tem grande utilização pelas empresas que produzem Banco de dados orientados a objeto. Possui sintaxe similar a SQL sendo de fácil compreensão e clareza. Além disso, a linguagem OQL também possui expressões de caminho na representação da navegação pelos relacionamentos.

Como desvantagens, pode-se citar a existência de muitos operadores, muitos recursos e várias opções de sintaxe, obrigando a adaptação da linguagem OQL ao Modelo Conceitual e definição de um subconjunto de operadores.

As consultas OQL não obedecem uma forma básica de construção como em SQL. Consultas válidas podem ser construídas, por exemplo, apenas com nomes ou métodos de objetos ou classes, por exemplo. Essa característica acaba criando novos desafios desnecessários de tradução.

2.2.4 Linguagens para navegação explícita - Álgebra para OQL

Álgebra para OQL [FEG 2001] possui expressões de caminho que expressam a navegação pelos relacionamentos, mas não é uma linguagem muito utilizada e conhecida. Também é de difícil entendimento. Sua aplicação é mais eficiente para otimização de consultas da linguagem OQL [FEG 2001a].

A seguir são apresentados dois exemplos de consulta para obter a soma da folha de pagamento dos empregados. No primeiro exemplo é feita a consulta na linguagem OQL da ODMG.

```
Sum(select e.salary from e in employees)
```

Nesta consulta é aplicado o método *sum*, que soma um conjunto de valores, sobre o atributo *salary* das instâncias de *employees* retornadas pela consulta entre parênteses.

Na álgebra para OQL a consulta equivalente é construída da seguinte forma:

```
reduce(sum,get(sum,Employes,e,and()),x,project(e,salary),and())
```

Nesta consulta, inicialmente é aplicada a projeção, para otimizar a consulta reduzindo a quantidade de informação manipulada. Após isto, é aplicado o método *sum* sobre o conjunto retornado.

2.2.5 Linguagens para navegação explícita - Linguagem de consulta LOREL

A linguagem de consulta LOREL faz parte de um projeto acadêmico [ABI 97] sendo uma derivação da linguagem OQL para consultas a base de dados semi-estruturados. Entre algumas vantagens do uso da mesma para consultas sobre o modelo conceitual estão a sintaxe simples e de fácil compreensão similar a SQL e OQL, e a representação da navegação pelos relacionamentos através de expressões de caminho.

Além disso o projeto da linguagem já é direcionado a Modelos Semi-estruturados, lidando com conjuntos heterogêneos, estruturados ou não; representações diferentes e irregulares da mesma informação e possível inexistência da representação de alguma informação.

A impossibilidade de se referenciar mais de um relacionamento entre conceitos é uma das desvantagens dessa linguagem. Algumas vezes o modelo conceitual necessita dessa representação para abstrair fontes XML heterogêneas.

A maior desvantagem desta linguagem, entretanto, é o fato de não ser um padrão consolidado. LOREL é uma linguagem pouco conhecida e testada sendo utilizada na sua maior parte apenas a nível acadêmico por um pequeno grupo de pesquisadores.

2.2.6 Linguagens para XML - Linguagem de consulta XPath 1.0 e 2.0

Como as consultas sobre o modelo conceitual serão traduzidas para XQuery 1.0, o uso da linguagem XPath 1.0 sobre o modelo conceitual possui a vantagem de dispensar conversões de sintaxe. Com isso, a construção do mecanismo de tradução

será simplificada. A linguagem XPath possui expressões de caminho e é uma linguagem padronizada pela W3C [XML 2001], além de ser largamente utilizada e testada.

A linguagem XPath 1.0 entretanto não permite a construção de uma saída com um novo formato, ou um novo documento. Diferentemente das linguagens analisadas anteriormente, XPath 1.0 é uma linguagem que permite apenas consultar partes de um documento ou suas características como ordem ou existência de elementos. Além disso, XPath também não permite mais de um relacionamento entre elementos, uma vez que sua estrutura hierárquica permite apenas o relacionamento *pai-filho* entre dois elementos. XPath possui operadores que trabalham sobre o paradigma hierárquico, exigindo conhecimento da estrutura do documento. Esses operadores não fazem sentido no escopo do modelo conceitual que não possui hierarquia nem noção de ordem entre os conceitos.

A nova versão da linguagem, a XPath 2.0 [XML 2002], definida em 2002 criou praticamente uma nova linguagem. Vários novos operadores foram adicionados com instruções de iteração e formatação de saída e foram definidos inúmeros tipos de dados específicos além do tipo padrão *text* presente em XPath 1.0.

De acordo com [LEN 2002], o objetivo de XPath 2.0 não é evoluir XPath 1.0. Numa tentativa do consórcio W3C de compartilhar esforços entre as linguagens XQuery 1.0 e XSLT 2.0 criou-se um subconjunto comum a estas duas linguagens, que foi chamado XPath 2.0. Várias características de XPath 2.0 e XQuery 1.0 são comuns.

Sendo assim na próxima seção será apresentado diretamente a linguagem XQuery 1.0, pois a linguagem XPath 2.0 isoladamente não trará novos recursos úteis ao presente trabalho.

2.2.7 Linguagens para XML - Linguagem de consulta XQuery 1.0

Como a linguagem de consulta definida para acesso as fontes XML é XQuery 1.0, o uso dessa linguagem também para construção das consultas sobre o modelo conceitual facilita o processo de tradução uma vez que não são necessárias traduções de sintaxe.

A linguagem de consulta XQuery é atualmente a linguagem mais utilizada para consulta a documentos XML. Esta linguagem foi projetada para ser de fácil leitura e compreensão possuindo construções que lembram SQL. A linguagem de consulta XQuery 1.0 é uma evolução de XPath e a construção das consultas XQuery utiliza expressões XPath. XQuery também é um Padrão definido pela W3C [W3C 2002].

A linguagem XQuery é muito mais poderosa que XPath 1.0. Pode-se, por exemplo, alternar rótulos e valores fixos, realizar iterações com partes de um documento consultado ou criar junções de vários outros documentos, gerando novos documentos de saída específicos para cada necessidade do usuário da consulta. XQuery 1.0 possui operadores que alimentam variáveis com elementos XML ou partes de documentos para posterior processamento, além de operadores de agregação, ordenação e atualização.

Entre as desvantagens, entretanto, está o fato que XQuery 1.0 também não permite mais de um relacionamento entre conceitos e possui operadores que exploram as características dos detalhes da estrutura de armazenamento. Soma-se a esses problemas o fato que a linguagem XQuery é extensa, com inúmeros outros operadores e construções de sintaxe, o que dificulta significativamente a criação do

mecanismo de tradução.

A linguagem XQuery é uma linguagem padronizada, entretanto está em permanente aperfeiçoamento assim como as outras linguagens definidas pelo W3C. A cada revisão, detalhes são melhorados e novos operadores são definidos deixando a linguagem cada vez mais extensa.

2.2.8 Definição da linguagem de consulta

O estudo apresentado é necessário para conhecer e analisar comparativamente as linguagens de consulta existentes na literatura a fim de que se possa definir a linguagem mais apropriada para construir consultas sobre o modelo conceitual.

Com essa análise verificam-se as vantagens e desvantagens de cada linguagem analisada segundo os critérios estabelecidos que nortearam a análise.

Segundo tais critérios, procura-se definir uma linguagem simplificada e que não obrigue o conhecimento dos detalhes da estrutura de armazenamento. Esses problemas são encontrados nas linguagens SQL e OQL, XPath e XQuery.

Outros problemas são verificados nas linguagens SQL e OQL. Estas linguagens são muito extensas e obrigam o conhecimento da estrutura de armazenamento. Além disso trabalham num paradigma muito diferente das fontes XML dificultando a construção do mecanismo de tradução.

As linguagens mais simples e abstratas, entretanto, como Álgebra OQL, Álgebra ER e Lorel, mostram outros problemas conforme já apresentado neste capítulo.

A linguagem XQuery é uma linguagem com muitos operadores e que se baseia no conhecimento da estrutura de armazenamento. Verifica-se, entretanto, a possibilidade de solucionar estes problemas com a criação de uma **nova linguagem** baseada em XQuery.

Com a criação de uma nova linguagem baseada em XQuery, o processo de tradução não necessita conversões de sintaxe e se torna menos complexo, uma vez que a linguagem de destino da tradução é a própria XQuery.

Dessa forma, optou-se pela criação de uma nova linguagem baseada em XQuery 1.0.

A nova linguagem de consulta criada chama-se *CXQuery* (conceptual XQuery) e contém modificações na XQuery original que visam sanar as lacunas ou problemas encontrados na análise realizada neste capítulo. Esta nova linguagem é apresentada no capítulo 3.

2.3 Mecanismos de tradução existentes

Recentemente, alguns trabalhos tentaram resolver o problema da construção de consultas sobre fontes XML heterogêneas. Esta seção apresenta alguns trabalhos que lidaram com os problemas de integração de esquemas e tradução de consultas.

2.3.1 Esquemas XML semânticos

Em [VID 2002] são utilizados esquemas XML semânticos que possuem estrutura hierárquica e representam o esquema de representação da informação das fontes. Há um mediador também hierárquico que representa a integração destes

esquemas semânticos. O trabalho se propõe não só a promover a integração de fontes XML entre si, mas também destas com esquemas relacionais. O mapeamento do mediador para os esquemas XML semânticos das fontes é feito utilizando um esquema de mapeamento baseado em assertivas de correspondência e relações de conjuntos.

A utilização de mediadores com esquemas hierárquicos podem criar problemas e complicações desnecessárias conforme apresentado no capítulo 2.1. Estes problemas são introduzidos pela estrutura hierárquica e não pelas consultas feitas [DAT 86]. Para tratar-los, o autor definiu chaves de referências para outras estruturas. Esse mecanismo acaba simulando de maneira nada elegante um esquema não hierárquico como o modelo conceitual proposto nesta dissertação.

A vantagem desse paradigma, segundo o autor, é a de poder usar a linguagem XPath 2.0 diretamente sobre o mediador, dispensando a necessidade da criação de uma nova linguagem. A tradução entretanto gera expressões XPath 2.0 inválidas, pois esta linguagem não possui operadores para tratar as chaves de referência inseridas na expressão pelo processo de tradução. O próprio autor confirma essa lacuna dizendo que a tradução gera expressões XPath que não podem ser utilizadas nas fontes, e afirma que as traduções criadas ainda devem passar por um outro processo para convertê-las em consultas XQuery válidas. Além disso, a tradução proposta em [VID 2002] traduz as consultas sobre o mediador para o paradigma dos esquemas semânticos locais e não para a estrutura relacional ou XML nativa das fontes.

Outro ponto pouco explicado é como utilizar XPath 2.0 em um modelo que represente um conjunto de fontes XML e relacionais. Operadores que acessam posições específicas do documento XML ou que trabalhem com ordem relativa entre os elementos tem um mapeamento complexo ou sem sentido para um modelo relacional. Mesmo sendo possível, a princípio as assertivas de correspondência propostas não tem como representar esses tipos de mapeamento. Operadores que acessam posições hierárquicas (filho, pai ,irmão) também não fazem sentido no modelo relacional e podem gerar resultados inconsistentes ou errados. A tradução de consultas XPath 2.0 que possuam operadores de iteração e de atribuição de valores à variáveis também não foi apresentada.

2.3.2 Esquema XML global

Em [MAN 2001] é proposta uma arquitetura que possibilita a construção de consultas sobre fontes XML heterogêneas ou fontes de dados relacionais. As consultas são feitas em XQuery da W3C sobre um esquema XML global que representa as fontes. Como esse esquema global é hierárquico, podem ocorrer as mesmas dificuldades de representação das relações vários-para-vários apresentadas no capítulo 2.1.

Após a construção dessas consultas XQuery, as mesmas são traduzidas para SQL, passando por várias etapas de tradução. Essas consultas SQL são usadas para consultar as fontes relacionais e as fontes XML. Nas fontes XML há ainda um passo adicional pois as consultas tem que ser traduzidas para uma linguagem de consulta XML.

Conforme mencionado em [MAN 2001], pode ser impossível traduzir muitas das consultas XQuery criadas nessa arquitetura.

Além disso, no caso da integração de fontes XML, as consultas XQuery são convertidas para SQL, um paradigma completamente diferente, e depois reconvertidas para retornar à uma linguagem nativa XML. Mesmo que possa se justificar esse

procedimento pela necessidade de promover a integração com as fontes relacionais e outras fontes XML, esse processo pode gerar perdas de consistência significativas nas consultas.

2.3.3 Considerações sobre as abordagens apresentadas

Verifica-se nas abordagens propostas em [VID 2002] e [MAN 2001] a necessidade de criar uma nova linguagem de consulta e fornecer ao usuário um modelo global não hierárquico. Do contrário, será difícil obter uma integração e tradução de consultas de forma concisa, abstrata e precisa das fontes de dados XML heterogêneas.

Neste capítulo foram apresentadas as razões para o uso do modelo conceitual para a integração de fontes XML e a necessidade da criação de uma nova linguagem de consulta. Esta nova linguagem será apresentada no próximo capítulo. Por fim, foi apresentada uma análise dos trabalhos relacionados ao escopo da presente dissertação.

3 Linguagem de consulta CXQuery

Neste capítulo será apresentada a linguagem de consulta CXQuery (Conceptual XQuery), que é baseada e possui uma sintaxe similar a XQuery 1.0 da W3C. Esta linguagem permite construir consultas sobre o modelo conceitual apresentado no capítulo 2.1 que é gerado na integração dos esquemas XML.

Os exemplos apresentados neste capítulo serão baseados no domínio de problema de um congresso já apresentado no capítulo 2.1. O esquema da fonte de número um, entretanto, foi estendido para permitir a representação da informação sobre os revisores dos artigos. As duas representações dos esquemas XML destas fontes são apresentados nas figuras 3.1 e 3.2.

Essa alteração no esquema da fonte 1 resultou uma alteração também no modelo conceitual global destas fontes. Este modelo conceitual pode ser visto na figura 3.3.

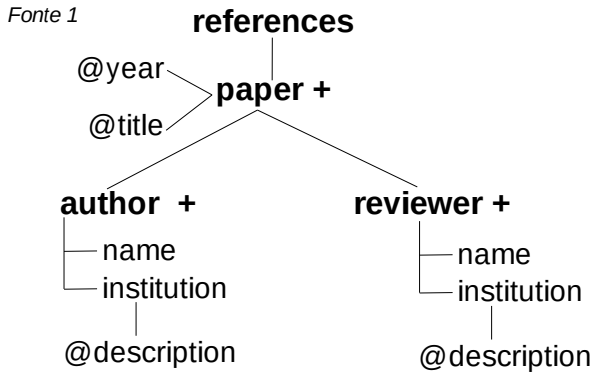


FIGURA 3.1 – Exemplo de fonte de dados XML - Fonte 1

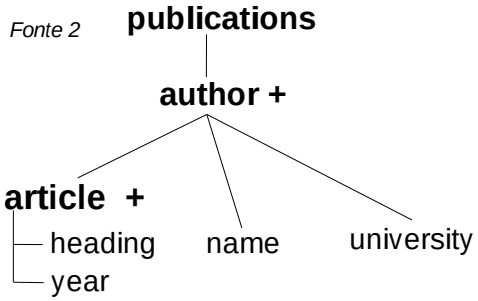


FIGURA 3.2 – Exemplo de fonte de dados XML - Fonte 2

Na fonte 1 mostrada na figura 3.1, o esquema é construído de forma que as publicações são representadas por uma coleção de elementos *paper* e cada *paper* possui todos os seus respectivos autores e revisores (elementos *author* e *reviewer*) como filhos, ou seja, em um nível hierárquico inferior. O ano de publicação é representado pelo atributo *year* e o título do artigo é representado pelo atributo *title*, ambos do elemento *paper*. O nome do autor ou revisor é representado pelo elemento *name* e o nome da instituição a que ele pertence está em um atributo *description* do elemento

institution. Tanto o elemento *name* quanto o elemento *institution*, estão abaixo do elemento *author* ou *reviewer* na hierarquia.

Já na fonte 2 mostrada na figura 3.2, o esquema é construído de forma que as publicações são representadas por uma coleção de elementos *author* ao invés de *paper*; e, cada *author* possui todos as suas respectivas publicações (desta vez como elemento *article*) como filhos. Nesta fonte estão ausentes informações sobre os revisores dos artigos. O título do artigo e o ano desta vez são representados por elementos ao invés de atributos, e com nomes *heading* e *year*. O nome do autor é representado através do elemento *name* e o nome da instituição está diretamente no elemento *university*.

O modelo conceitual que representa as duas fontes apresentadas nas figuras 3.1 e 3.2 pode ser visto na figura 3.3.

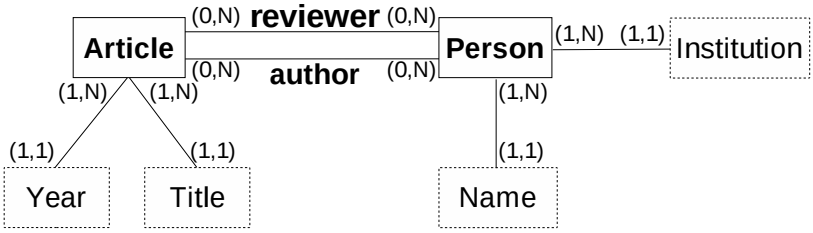


FIGURA 3.3 – Modelo Conceitual

No modelo conceitual da figura 3.3 há os conceitos *year* e *title* que estão relacionados com o conceito *article*, representando o ano e o título do artigo.

Os conceitos *name* e *institution* estão relacionados com o conceito *person* e representam o nome de uma pessoa e a instituição a que a mesma pertence.

As funções desempenhadas pelas pessoas sobre os artigos são representados pelos relacionamentos *reviewer* e *author*. O relacionamento das instâncias de pessoas com seus respectivos artigos escritos, é obtido pela navegação no relacionamento *author* e o relacionamento das instâncias de pessoas e seus respectivos artigos revisados, são obtidas pela navegação no relacionamento *reviewer*. Maiores detalhes sobre o uso dos relacionamentos do modelo conceitual serão apresentados no capítulo 3.1.

A fim de organizar a tarefa de definição e tradução da linguagem CXQuery, foi construída primeiramente uma linguagem de expressões de caminho para expressar a navegação e acesso às instâncias dos conceitos do modelo conceitual. Essa linguagem é uma modificação da linguagem XPath 1.0 e foi denominada *CXPath (Conceptual XPath)*.

Segundo [W3C 2002b] as linguagens XQuery 1.0 e XSLT 2.0 podem ser consideradas linguagens hospedeiras das expressões XPath. Uma consulta XQuery utiliza expressões XPath para expressar a navegação e acessar os elementos e atributos no documento XML.

Dessa forma, a linguagem CXQuery(Conceptual XQuery)também é a linguagem hospedeira das expressões CXPath(Conceptual XPath). Dentro de uma consulta CXQuery podem ser utilizadas inúmeras expressões CXPath para expressar o acesso às instâncias dos conceitos.

3.1 A linguagem CXPath

A seguir serão apresentadas as características da linguagem de consulta CXPath(Conceptual XPath), as modificações definidas e as diferenças em relação a linguagem XPath 1.0.

- Nodo raiz

  O modelo conceitual é um grafo genérico, não uma árvore. Dessa forma, não possui noção de nodo raiz. Uma expressão CXPath absoluta pode iniciar em qualquer conceito do modelo, e, a partir deste, navegar nos relacionamentos existentes em qualquer direção.

  A sintaxe de uma expressão CXPath é muito semelhante a sintaxe de uma expressão XPath 1.0.

  Nas expressões XPath 1.0 a expressão de caminho navega em direção aos elementos ou atributos dos relacionamentos hierárquicos. Já em CXPath as expressões de caminho navegam pelos relacionamentos em direção aos conceitos do modelo conceitual.

  No exemplo a seguir, a consulta CXPath retorna todos os títulos.

  ```
  /title
  ```

  Esta consulta inicia no conceito *title* e retorna todas as instâncias do conceito *title*.

  No próximo exemplo a consulta CXPath solicita "Todos os títulos de artigos":

  ```
  /article/title
  ```

  Esta consulta inicia no conceito *article* e navega no relacionamento existente com o conceito *title*. Dessa forma retorna todos os títulos de artigos existentes. Esse resultado pode ou não ser o mesmo do primeiro exemplo. A consulta deve ser construída usando o conceito article para garantir o retorno apenas das instâncias de títulos que possuem relacionamento com artigos.

  Outro exemplo é uma consulta para obter "Todos os nomes de pessoas". Esta consulta é apresentada abaixo:

  ```
  /person/name
  ```

  A consulta inicia no conceito *person* e navega no relacionamento para alcançar o conceito *name* relacionado e assim obter todos os nomes de pessoas.

  A consulta sem conceitos, apenas com o operador de navegação "/"não é válida em CXPath.

- Acesso à conceitos

  Uma expressão de caminho CXPath navega pelos relacionamentos do modelo conceitual para atingir as instâncias dos conceitos desejados. Como CXPath não possui relacionamentos direcionados e hierárquicos, foi estabelecida uma nova função semântica para o operador de navegação "/". Em XPath 1.0 ele representa a navegação no relacionamento hierárquico "pai-filho". Já em CXPath, o operador "/"representa o caminho em direção às instâncias de um conceito através de um relacionamento. O acesso as instâncias dos conceitos desejados é feita de forma equivalente ao acesso aos elementos em XPath 1.0.

- Operador de seleção

  O operador de seleção "[ ]"em CXPath assume a mesma função existente em XPath 1.0.

  A seguir é apresentado um exemplo de uso do operador de seleção. Neste exemplo é construída uma consulta CXPath para obter "Títulos dos artigos produzidos em 1999".

  ```
  /article[year = "1999"]/title
  ```

  A consulta inicia no conceito *article*. Somente os artigos que estiverem relacionados com as instâncias do conceito *year* de valor "1999"são selecionados. Por fim, navega-se até o contexto do conceito *title*. Dessa forma, são retornadas todas as instâncias do conceito *title* relacionadas com as instâncias do conceito *article* filtradas pelo critério de seleção.

- Acesso à atributos

  A sintaxe de acesso a atributos não está presente em CXPath uma vez que tanto elementos quanto atributos semanticamente equivalentes das fontes XML são representados por conceitos.

- Operadores hierárquicos

  O modelo conceitual resultante da integração das fontes XML representa a navegação pelos relacionamentos de maneira abstrata em um grafo sem sentido de navegação definido (não hierárquico).

  Por conseqüência, as expressões CXPath não possuem os operadores que exploram direta ou indiretamente características hierárquicas como descendência e ascendência presentes em XPath e XQuery. Como exemplo pode-se citar os operadores *ancestor*,*child*,*descendant*,*"..."*, entre outros.

  O operador *"/"* conforme já mencionado, foi mantido. Este operador, entretanto, possui uma semântica diferente de XPath 1.0 onde promove a busca do elemento hierarquicamente descendente. Em CXPath o operador "/"representa a navegação genérica pelos relacionamentos do modelo conceitual.

- Informação de ordenamento dos elementos

  A utilização direta de operadores que exploram a informação de ordenamento dos conceitos sobre o modelo conceitual pode produzir interpretações inconsistentes em uma consulta CXPath.

Como o modelo conceitual representa um conjunto de fontes XML, as fontes podem possuir informações diferentes implícitas na noção de ordem de seus elementos. Uma consulta sobre o modelo conceitual que fosse em busca da terceira ocorrência de um elemento, por exemplo, poderia gerar um resultado inconsistente ou fora das expectativas do construtor da consulta.

Em determinadas fontes XML a ordem de ocorrência dos elementos no documento pode representar a cronologia da criação dos elementos. Já em outras fontes a ordem de ocorrência dos elementos pode representar outras informações como ordenamento por faixa etária, ordenamento de prioridade, ou outras noções de ordem.

Em uma consulta integrada poderá ser retornado, por exemplo, o terceiro elemento mais prioritário de uma fonte, o terceiro elemento por ordem de chegada de outra fonte e o terceiro elemento que possua a terceira menor informação de idade associada. Este resultado pode ser inútil na maioria das vezes, pois associa a noção de ordem em informações que não são do mesmo tipo.

O problema de lidar com a integração e a tradução correta da noção de ordem dos elementos XML não faz parte do escopo deste trabalho.

CXPath, por conseqüência, não possui operadores que tratam da ordem de armazenamento dos elementos ou acesso a elementos através de posicionamentos específicos dentro das fontes. Exemplos que podem ser citados são as construções com operadores *position*, *last*, *first*, entre outros.

- Operadores coringa

  Em XPath 1.0 existem operadores como *"//"* e *"*"*, que funcionam como *caracteres coringa*, dispensando partes da informação do caminho a ser percorrido.

  O modelo conceitual é um grafo e pode apresentar vários caminhos possíveis para se atingir um determinado conceito, podendo inclusive possuir ciclos. Como operadores coringa irão percorrer todos os caminhos possíveis entre dois contextos determinados, podem ocorrer casos em que infinitos caminhos existam pela composição de ciclos nas expressões possíveis.

  Para se evitar estes problemas, operadores como estes não são permitidos na linguagem CXPath.

- Relacionamentos

  Um documento XML possui apenas um relacionamento entre dois elementos ou atributos: o relacionamento hierárquico *pai-filho*.

  O modelo conceitual, entretanto, pode apresentar mais de um relacionamento entre os conceitos. Isto ocorre quando dois conceitos possuem entre si mais de um relacionamento quando vistos do ponto de vista de suas fontes na criação do modelo conceitual. Nesses casos os relacionamentos devem ser nomeados para evitar ambigüidade.

  Para dar suporte a isso foi criada nas expressões CXPath uma modificação que identifica por qual dos relacionamentos a consulta está navegando. Para isto, antes do nome do conceito destino da navegação, é identificado entre chaves o

nome do relacionamento que está sendo utilizado. Isto pode ser exemplificado na consulta *"Nome dos autores de artigos revisados por Ivo Silva"* da figura 3.4

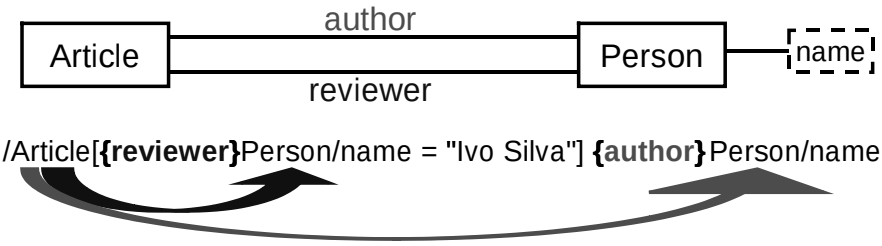


FIGURA 3.4 – Consulta com navegação por relacionamentos nomeados

Quando há apenas um relacionamento, a colocação de nome é opcional no modelo conceitual. Neste caso, a navegação das expressões CXPath é conforme apresentado anteriormente no relacionamento entre os conceitos *article* e *title*, onde o processo de integração optou pela não identificação do nome do relacionamento no modelo conceitual.

- Papéis

  Outro acréscimo proposto pelas expressões CXPath é a possibilidade de se referenciar papéis aos conceitos participantes de um relacionamento. Em alguns casos isto se faz necessário, principalmente quando se tem auto-relacionamentos.

  Em auto-relacionamentos não é possível identificar qual o papel que uma determinada instância do conceito está assumindo. Em uma consulta CXPath sobre o modelo conceitual da figura 3.5, por exemplo, não é possível distinguir os nomes dos maridos ou das esposas se apenas o nome do relacionamento *casamento* estiver presente. Isto ocorre pois o mesmo conceito representa genericamente os tipos de instâncias de ambos os lados de um auto-relacionamento.

  Sendo assim, a sintaxe das expressões CXPath proposta apresenta a possibilidade de se expressar os papéis quando estes aparecem em determinados conceitos do modelo conceitual.

  Para especificar as instâncias com um papel específico em um conceito, após o nome do relacionamento escreve-se o nome do papel que se deseja acessar do conceito destino. Dessa forma, uma consulta para obter o nome da esposa de "José da Silva", será formulada da seguinte forma:

```
Pessoa[Name="José da Silva"]/{casamento.esposa}Pessoa.Name
```

Como pode ser observado, as expressões CXPath possuem várias modificações para adequá-las ao paradigma do modelo conceitual. Essas modificações são formalmente definidas na gramática CXQuery anexa no capítulo 5 a partir da produção do símbolo não terminal *CXPathExpr*, que representa a análise das expressões CXPath dentro de uma consulta CXQuery. Na próxima seção será apresentada a linguagem de consulta CXQuery e outros exemplos do uso das linguagens CXPath e CXQuery.

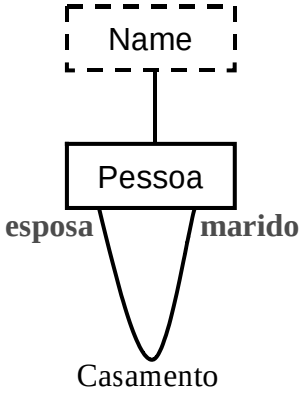


FIGURA 3.5 – Modelo conceitual parcial mostrando o uso de papéis

3.2 A linguagem CXQuery

A seguir serão apresentados as características da linguagem de consulta CXQuery(Conceptual XQuery), que permite a construção de consultas sobre o modelo conceitual global.

Um dos objetivos na escolha da linguagem mencionada no capítulo 2.2 foi a de escolher uma linguagem compacta com os operadores básicos, para não tornar a tarefa de construção do mecanismo de tradução muito extensa, complexa ou até inviável.

Sendo assim, devido ao grande número de possíveis construções de consulta existentes na linguagem XQuery como pode ser observado em [W3C 2002a] e [W3C 2003], limitou-se neste trabalho as construções possíveis na linguagem CXQuery. As construções possíveis serão apresentadas a seguir.

Dentro das consultas CXQuery, podem ser utilizadas consultas CXPath para acesso e navegação pelas instâncias dos conceitos, conforme apresentado anteriormente.

- Retorno dos dados

  Uma característica importante da linguagem CXQuery diz respeito ao resultado das consultas. Na linguagem XQuery 1.0 da W3C pode-se construir uma consulta para retornar tanto elementos léxicos (elementos "folha"ou atributos que possuem um valor diretamente associado) quanto elementos não-léxicos (elementos compostos por outros elementos).

  Em CXQuery, entretanto, os conceitos não-léxicos representam elementos compostos que podem possuir estruturas muito heterogêneas em cada uma das fontes XML , e isso pode ocasionar resultados incompreensíveis em uma solicitação de retorno não léxica. Numa resposta de consulta podem estar misturados, por exemplo, elementos com seqüências de elementos PCData e seqüências de grandes elementos compostos, retornando vários outros dados.

  Essas diferenças estruturais nos conceitos não-léxicos também podem afetar resultados intermediários, gerando resultados finais imprevisíveis, sem sentido ou até com erro.

  Isto pode ocorrer nos operandos dos critérios de seleção da consulta que produzem resultados sem sentido ao compararem valores em conceitos não-léxicos que possuam uma estrutura incompatível uma em relação a outra.

Para lidar com esse problema e permitir o retorno não-léxico nas consultas da linguagem, devem ser construídos mecanismos para análise e integração das instâncias dos resultados, o que está fora do escopo da presente dissertação. Esses mecanismos podem uniformizar ou criar padrões compreensíveis no retorno dos dados.

Sendo assim, definiu-se que consultas válidas na linguagem CXQuery permitem apenas retornar conceitos léxicos do modelo conceitual.

As expressões CXPath dentro de uma consulta CXQuery também não podem retornar elementos não-léxicos. Isso não é verificado pela gramática, mas sim pelo mecanismo de tradução gerando um erro em tempo de tradução. A única exceção é no contexto da variável de iteração do operador *for* que pode e deve retornar ambos os tipos de conceitos para o permitir a varredura de todas as suas instâncias.

- Construtores de elementos XML

  O uso de construtores de elementos XML em CXQuery segue a mesma sintaxe de XQuery 1.0.

  Dessa forma, uma consulta CXQuery pode ser construída com ou sem construtores de elementos XML. Eles são úteis na formatação do documento de resultado da consulta. Nesse caso o corpo da consulta CXQuery deve estar dentro dos delimitadores de início e fim de consulta: "{"e "}"e os construtores do documento, do lado de fora.

  Se a consulta CXQuery for declarada sem o uso de construtores XML, estes delimitadores não são necessários.

- Operador *for*

  Ao iniciar uma consulta CXQuery é possível escrever uma expressão CXPath absoluta, ou utilizar uma ou várias construções do tipo "*for <variável de iteração > in <contexto>*", iterando instâncias de conceitos em várias variáveis de iteração diferentes. O *contexto* a ser iterado deve ser escrito em CXPath.

- Operador *where*

  Após a construção com o operador *for* pode ser usado construções com os operadores *where* ou diretamente o *return*, dependendo se o usuário deseja ou não filtros. Dessa forma, é opcional a construção *"where <expressão CXPath de iteração> <operador de comparação> <critério de comparação>"*.

- Operador *return*

  A construção com o operador *return* segue a sintaxe *"... return (<construções XML> <expressão CXPath de iteração> <construções XML>)*"*. Dentro do argumento do operador *return* também é possível definir construções XML. Assim como XQuery da W3C, estas construções irão aparecer em cada escrita de instância que estiver sendo iterada.

- Variáveis de Iteração

O funcionamento das variáveis de iteração em CXQuery é semelhante ao funcionamento em XQuery 1.0. Ou seja, todas as instâncias do contexto da variável de iteração são varridas, permitindo acesso às instâncias dos conceitos relacionados.

São estas as construções possíveis na linguagem CXQuery, maiores detalhes sintáticos podem ser conferidos na gramática anexa.

A princípio vários outros operadores de XQuery 1.0 poderiam fazer parte da linguagem CXQuery. Exceção seria feita aos operadores que exploram características do paradigma hierárquico ou de detalhes de armazenamento. Optou-se, entretanto, por um conjunto de operadores básicos para ficar em conformidade com características esperadas da linguagem, definidas no capítulo 2.2. Com isso, a tarefa de construção do mecanismo de tradução será menos extensa e complexa.

3.3 Exemplos de consultas CXQuery

A seguir serão apresentados exemplos de consulta CXQuery.

3.3.1 Exemplo 1

As construções CXQuery necessitam utilizar dentro de seus operadores as expressões CXPath. As funções principais desempenhadas pelas expressões CXPath são a navegação pelos relacionamentos e acesso às instâncias dos conceitos.

Além disso, uma expressão CXQuery pode ser simplesmente uma única expressão CXPath. O exemplo abaixo solicita em CXPath "Todos os nomes de autores de artigos":

```
/article/{author}person/name
```

Para retornar apenas os autores (e não os revisores), a consulta navega pelo relacionamento de autoria entre os artigos e as pessoas.

3.3.2 Exemplo 2

Podem ser usados construtores de elementos XML para criar um novo documento de saída. Como exemplo deseja-se retornar "O nome de todas as pessoas"dentro do elemento *everybody*:

```
<everybody>
{
/person/name
}
</everybody>

ou

<everybody>
{
/article/{author}person/name
```

```
/article/{reviewer}person/name
}
</everybody>
```

Na primeira opção a consulta é iniciada pelo conceito *person*. O conceito léxico *person* possui um relacionamento com o conceito *name*. Assim são retornados todos os nomes das pessoas, tanto autores como revisores.

Na segunda opção o usuário preferiu iniciar a consulta pelo conceito *article*. Nessa opção, para chegar ao conceito *person* e retornar tanto os autores quanto os revisores, devem ser construídas duas expressões CXPath. Em uma das expressões, a navegação é feita pelo relacionamento *author*, retornando os nomes de todos os autores. Na outra expressão, a navegação é feita pelo relacionamento *reviewer* retornando os nomes de todos os revisores. Com as duas expressões obtém-se o mesmo resultado da primeira opção de consulta.

3.3.3 Exemplo 3

Neste exemplo de consulta CXQuery deseja-se obter os títulos dos artigos escritos por "Juvenal Antunes":

```
for $i in /article
where $i/{author}person/name = "Juvenal Antunes"
return $i/title
```

O operador *for* varre todas as instâncias retornadas pela expressão absoluta CXPath */article* ligando cada instância do conceito *article* a variável i. Usando a variável de iteração, cada instância do conceito é testada na cláusula *where* onde é definido que apenas os artigos escritos por "Juvenal Antunes"devem ser retornados.

Finalmente na cláusula *return* é retornado os títulos relacionados às instâncias dos conceitos filtrados na cláusula *where*.

3.3.4 Exemplo 4

A consulta a seguir constrói um novo documento XML na saída cujo elemento raiz é *result*. Dentro desse elemento haverá um elemento *article_after_97* que conterá os artigos escritos após o ano de 1997 e um elemento *all_reviewers_and_authors* que conterá todos os integrantes do congresso.

Os títulos dos artigos escritos após 1997 e respectivos autores serão apresentados nos elementos *title_name* e *author* dentro do elemento "pai"*article_after_97*.

Após apresentar todas as instâncias que compõe o elemento *article_after_97* será retornado o elemento *all_reviewers_and_authors* que apresentará todos os nomes de integrantes do congresso, tanto revisores quanto autores.

Nesta consulta há duas construções *"for"* independentes, a primeira liga a variável *ar* à cada iteração das instâncias do conceito *article* e a segunda liga a variável *per* à cada iteração das instâncias do conceito *person*

Apenas nas instâncias dos conceitos da variável *ar* é aplicado o critério de seleção da cláusula *where*.

Na cláusula *return* as instâncias dos conceitos associados a cada iteração das variáveis de iteração são relacionados aos conceitos léxicos de saída conforme a navegação pela expressão CXPath relativa.

```
<result>
{
for $ar in /article
for $per in /person
where $ar/year >= "1997"
return
<article_after_97>
    <title_name>
        $ar/title
    </title_name>
    <author>
        $ar/{author}person/name
    </author>
</article_after_97>
<all_reviewers_and_authors>
    $per/name
</all_reviewers_and_authors>
}
</result>
```

Um documento retornado após a aplicação desta consulta traduzida sobre as fontes, tem a seguinte forma:

```
<result>
    <article_after_97>
        <title_name>
            artigo x
        </title_name>
        <author>
            nome x
        </author>
        <title_name>
            artigo y
        </title_name>
        <author>
            nome y
        </author>
        ...
    </article_after_97>
    <all_reviewers_and_authors>
        nome x
        nome y
```

```
        nome z
        nome w
        ...
    </all_reviewers_and_authors>
</result>
```

3.3.5 Exemplo 5

Os critérios de seleção das expressões CXPath podem possuir novas expressões CXPath aninhadas em um novo contexto, conforme a consulta "Nome dos autores de artigos que foram escritos no mesmo ano do artigo de nome XML Overview":

```
/Article[Year=/Article[Title ="XML Overview"]/Year]/{author}person/name
```

Na expressão interna */Article[Title = "XML Overview"]/Year* é retornado o ano que o artigo em questão foi escrito, para fins desta explicação aqui denominado *XXXX*. A expressão CXPath externa recebe o parâmetro e processa a consulta em seu contexto já com o resultado obtido: */Article[Year = XXXX]/{author}person/name* e assim devolve o nome do autor de artigos escritos no ano *XXXX*.

Este capítulo apresentou a linguagem CXQuery desenvolvida a partir das linguagens XPath 1.0 e XQuery 1.0 da W3C para possibilitar consultas sobre o Modelo Conceitual que representa uma visão integrada de fontes XML. Estas consultas CXQuery devem agora ser traduzidas para uma linguagem de consulta nativa das fontes de dados XML. Com isso será possível o acesso aos dados dessas fontes.

4 Tradução de consultas

Neste capítulo é apresentado o mecanismo de tradução das consultas CXQuery para XQuery 1.0. Além disso, é apresentado a abordagem de mapeamento dos conceitos e relacionamentos do modelo conceitual para as respectivas construções nas fontes de dados XML. O mecanismo de tradução acessa essa informação de mapeamento para efetuar a tradução. O mecanismo de tradução está implementado segundo a arquitetura apresentada na figura 4.7.

Os exemplos deste capítulo serão baseados no domínio de problema apresentado no capítulo 3. As fontes XML são apresentadas nas figuras 3.1 e 3.2 e o respectivo modelo conceitual é apresentado na figura 3.3.

4.1 Informação de mapeamento

Na literatura de integração de banco de dados, duas abordagens são normalmente empregadas para definir a informação de mapeamento: Esquema de mapeamento e Visões [ELM 99].

- Esquema de Mapeamento - Banco de dados auxiliar com as informações de mapeamento. Essa abordagem funciona como um dicionário, e permite uma combinação direta da informação com sua respectiva tradução. Facilita a construção de mecanismo de tradução mas em alguns casos não é suficiente para tradução das consultas. Isso ocorre nos casos em que para se atingir a tradução de um conceito tenha que se navegar por várias direções de relacionamentos nas fontes, utilizar funções de alguma linguagens de consulta ou fazer seleções e combinações de vários dados para se obter a tradução.

- Visões - Uma visão descreve como os conceitos são definidos nos esquemas locais. As visões dão mais poder de expressão à tradução pois permitem a construção de consultas para representar a informação a ser mapeada. Em alguns casos, entretanto, isso pode tornar complexa a definição das visões.

Quanto maior for a heterogeneidade entre as fontes, mais complexo será o processo de integração e os mapeamentos gerados para se atingir um modelo conceitual comum.

Com isto será necessário utilizar funções, seleções, navegações em todas direções e outros processamentos possíveis dentro das fontes XML para se obter as correspondências desejadas. Nesse sentido, optou-se por expressar os mapeamentos através de visões das fontes XML.

O próximo desafio é definir também a linguagem de definição das visões. Um dos problemas da abordagem de mapeamento por visões é a complexidade para expressar determinadas visões.

Para não tornar muito complexa a definição das visões optou-se por uma linguagem simples e que fosse contida na linguagem XQuery para facilitar a construção do mecanismo de tradução. A linguagem escolhida para este fim foi XPath 1.0 [XML 2001].

Sendo assim, cada conceito ou relacionamento do Modelo Conceitual foi definido utilizando-se visões XML em termos dos elementos e atributos das bases XML.

Um determinado elemento/atributo pode ocorrer em diversos pontos da hierarquia do esquema da fonte XML, em diferentes contextos, como por exemplo, *endereço* do autor e *endereço* da conferência. XPath fornece uma *busca não ambígua*, que identifica as ocorrências deste elemento/atributo dentro do contexto desejado.

Como as fontes XML possuem estrutura de um grafo hierárquico, uma *expressão de caminho* que parta de um dos elementos raiz da DTD até o elemento/atributo desejado é suficiente para localizá-lo de forma precisa. Prova disto é que expressões de caminho são o princípio básico das pesquisas empregadas por linguagens de consulta para dados XML [W3C 2002].

Serão apresentados a seguir os dois tipos de mapeamento definidos, mapeamentos de conceitos e de relacionamentos.

4.1.1 Informação de mapeamento dos conceitos do modelo conceitual

A informação de mapeamento dos conceitos do modelo conceitual indica como o elemento ou atributo correspondente a este conceito é alcançado na fonte XML a partir do elemento raiz do documento. O *mapeamento de um conceito* do modelo conceitual para uma fonte XML é definido como uma *expressão de caminho absoluta XPath*, ou seja, um caminho que indica a hierarquia de elementos que deve ser percorrida a partir da raiz do documento até o elemento ou atributo desejado. Uma expressão de caminho absoluta em XPath inicia com o símbolo "/". Um exemplo desse mapeamento pode ser visto na figura 4.1 onde o conceito *article* do modelo conceitual é mapeado para as respectivas instâncias equivalentes ao conceito em cada fonte nos termos dos elementos locais através de expressões XPath 1.0.

Pode se observar que o caminho utilizado para se chegar nas instâncias equivalentes ao mesmo conceito varia bastante entre os esquemas das fontes. A técnica de mapeamento através de visões XPath, dessa forma, foi eficaz na geração das correspondências corretas.

4.1.2 Informação de mapeamento dos relacionamentos do modelo conceitual

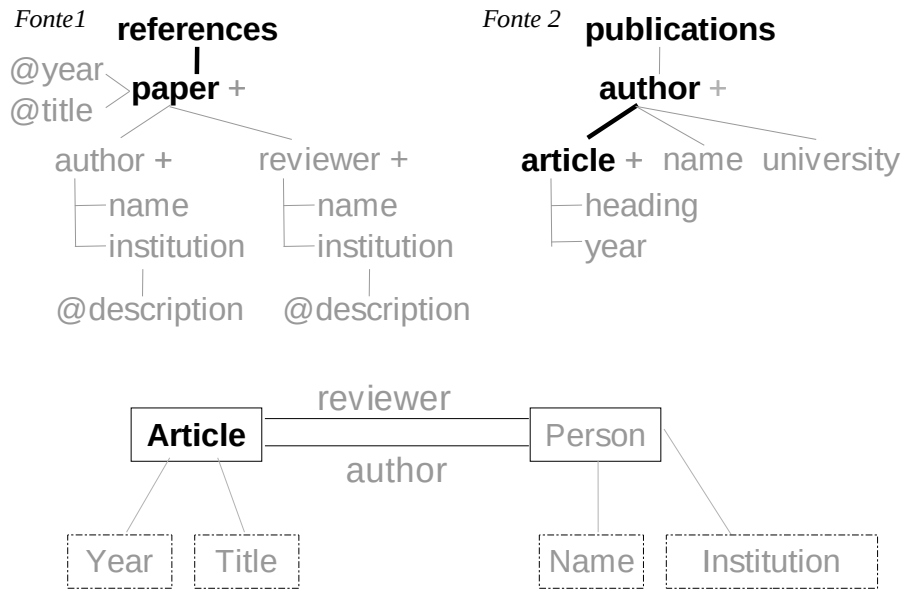
A informação de mapeamento dos relacionamentos do modelo conceitual indica como alcançar um conceito relacionado a outro conceito através de um caminho existente na fonte XML. O *mapeamento de um relacionamento* do modelo conceitual é representado através de uma *expressão de caminho relativa XPath*.

Esta expressão XPath indica como alcançar um elemento/atributo correspondente a um conceito destino a partir de um elemento/atributo correspondente a um conceito origem na fonte XML. O mapeamento é definido em ambos os sentidos do relacionamento para facilitar a tradução de consultas globais, que podem percorrer o modelo conceitual em qualquer direção.

Essa característica é relativamente eficaz para que a tradução da consulta CXQuery, feita sem o conhecimento da estrutura das fontes, seja corretamente mapeada, utilizando os caminhos e direções existentes e possíveis em cada fonte XML.

Quando o modelo conceitual, e conseqüentemente a consulta CXQuery, explicitarem nomes de relacionamentos e papéis, todas as opções possíveis de navegação entre dois conceitos utilizando os relacionamentos e/ou papéis devem ser mapeadas.

Essa dissertação aborda apenas as técnicas e abordagens de mapeamento utilizadas e definidas para possibilitar a armazenagem da informação de mapeamento.



Conceito/Relacionamento	*Nº da Fonte*	*Mapeamento*
Article	1	/references/paper
Article	2	/publications/author/article

FIGURA 4.1 – Exemplo que mostra o mapeamento armazenado de um conceito

Conforme mencionado anteriormente, a efetiva alimentação das informações de mapeamento é responsabilidade do mecanismo de integração, o qual deve gerar a informação de mapeamento a medida que integra as fontes XML heterogêneas, gravando as correspondências criadas usando as técnicas e abordagens de mapeamento definidas neste capítulo.

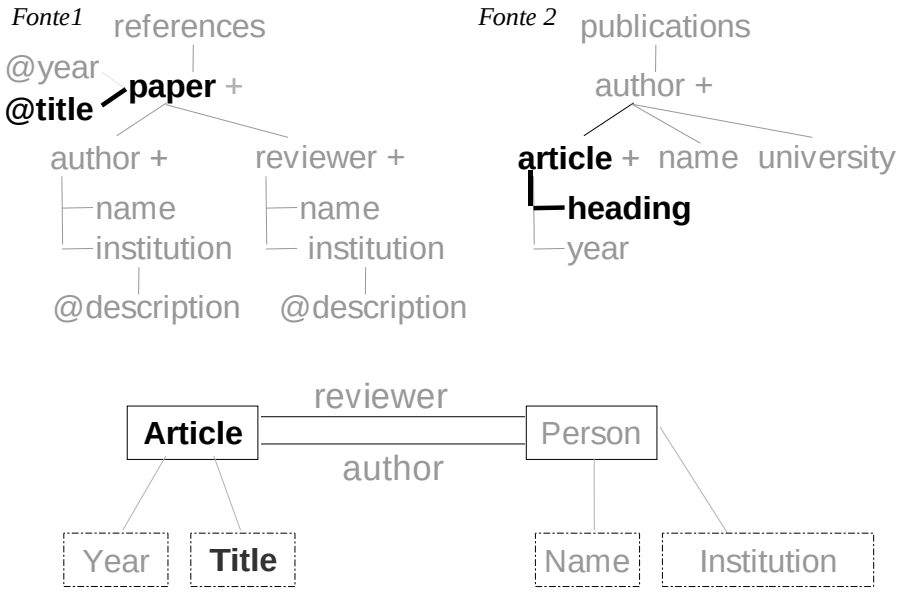
Na figura 4.2 temos um exemplo do mapeamento de um relacionamento existente entre o Conceito *article* e o conceito *title*. Na fonte 1 o elemento equivalente ao conceito *article* é *paper*, e ele será o nodo de contexto. Para se chegar a partir do nodo contexto ao conceito *title*, que na fonte é o atributo *title*, devemos utilizar a expressão *@title*.

Seguindo esse raciocínio, na fonte 2, o título do artigo é um elemento ao invés de um atributo e possui outro nome. Assim o expressão XPath relativa será *heading* ao invés de *@title*.

A expressão de caminho do mapeamento é basicamente um meio de, saindo do nodo contexto, através de um relacionamento/papel chegar em um outro conceito. Esse caminho relativo mostra sua utilidade no processo de tradução que é mostrado mais adiante.

Na figura 4.3 temos um exemplo de um mapeamento mais complexo do relacionamento *author* existente do conceito *Person* em direção ao conceito *Article*.

Neste exemplo, na fonte 2 o mapeamento é trivial, bastando-se descer um nível na hierarquia buscando o elemento descendente *article*. Já na fonte 1 deve-se primeiro garantir que o nodo contexto seja o elemento *author*, pois o mapeamento solicitado é pelo relacionamento de autoria e o conceito *person* pode ser tanto *reviewer* como *author*. Isso é feito com a utilização da função *local_name* conforme



Conceito/Relacionamento	*Nº da Fonte*	*Mapeamento*
Article → Title	1	@title
Article → Title	2	heading

FIGURA 4.2 – Exemplo de um mapeamento simples de um relacionamento

apresentado. Após isso, ao contrário da fonte 2 onde buscou-se o elemento descendente, na fonte 1 deve-se buscar o elemento ascendente pois a direção da navegação da fonte 1 é inversa a fonte 2. Isso é feito com o operador ".."de XPath 1.0. Por fim o mapeamento é representado pela seguinte expressão XPath: *[local_name(.) = "author"]/..*

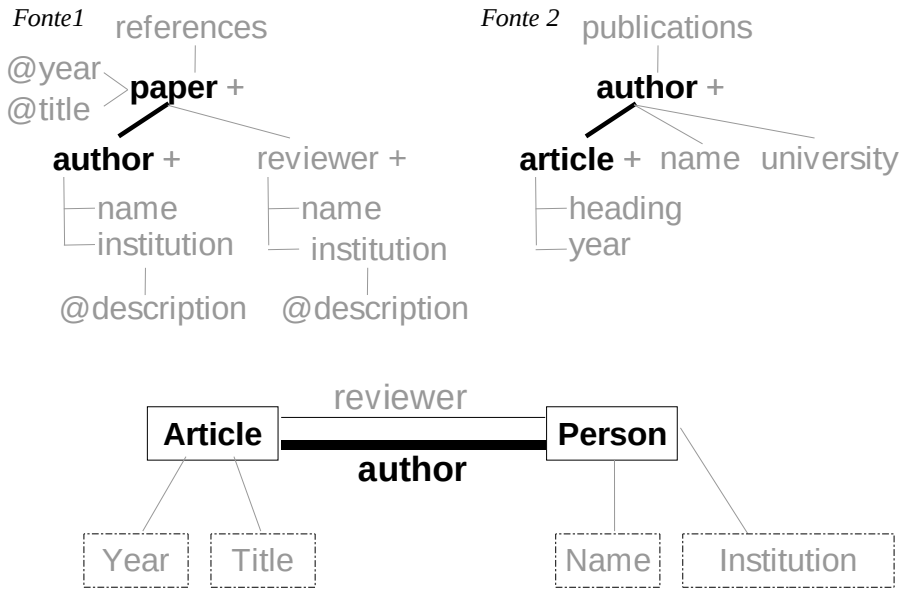
Na tabela 4.1 é apresentada a informação de mapeamento armazenada para as fontes 1 e 2 de todos os conceitos e relacionamentos do modelo conceitual da figura 3.3. Este mapeamento apresenta em cada tupla o nome do conceito ou relacionamento do modelo conceitual, o identificador da fonte XML e a visão XPath 1.0 que mapeia a informação conceitual para esta fonte XML.

O campo *ID_modelo* identifica de qual modelo conceitual a informação está sendo mapeada, pois mapeamentos de outros modelos conceituais podem ser armazenados nesse repositório. Há dois tipos de mapeamentos definidos, mapeamento de conceitos e mapeamento de relacionamentos. A coluna *tipo_mapeamento* identifica qual é o tipo de mapeamento que foi armazenado em cada tupla.

A definição da representação da informação de mapeamento apresentada neste capítulo foi desenvolvida em conjunto com [MEL 2000] que definiu a arquitetura de integração da fontes e a criação do modelo conceitual. Essa arquitetura é responsável pela alimentação da informação de mapeamento que é utilizada pelo mecanismo de tradução das consultas.

TABELA 4.1 – Informação de mapeamento armazenada

conceito/relacionamento	fonte	mapeamento	ID_Modelo	tipo_mapeamento
article	2	/publications/author/article	dissertacao	conceito
name	2	/publications/author/name	dissertacao	conceito
person	2	/publications/author	dissertacao	conceito
title	2	/publications/author/article/heading	dissertacao	conceito
year	2	/publications/author/article/year	dissertacao	conceito
article/{author}person	2	..	dissertacao	relacionamento
article/{reviewer}person	2		dissertacao	relacionamento
article/title	2	heading	dissertacao	relacionamento
article/year	2	year	dissertacao	relacionamento
name/person	2	..	dissertacao	relacionamento
person/{author}article	2	article	dissertacao	relacionamento
person/{reviewer}article	2		dissertacao	relacionamento
person/name	2	name	dissertacao	relacionamento
title/article	2	..	dissertacao	relacionamento
year/article	2	..	dissertacao	relacionamento
institution	2	/publications/author/university	dissertacao	conceito
person/institution	2	university	dissertacao	relacionamento
institution/person	2	..	dissertacao	relacionamento
person/institution	1	institution/@description	dissertacao	relacionamento
institution	1	/references/paper/(author|reviewer)/institution/@description	dissertacao	conceito
article	1	/references/paper	dissertacao	conceito
person	1	/references/paper/(author|reviewer)	dissertacao	conceito
title	1	/references/paper/@title	dissertacao	conceito
year	1	/references/paper/@year	dissertacao	conceito
name	1	/references/paper/(author|reviewer)/name	dissertacao	conceito
title/article	1	..	dissertacao	relacionamento
year/article	1	..	dissertacao	relacionamento
name/person	1	..	dissertacao	relacionamento
article/title	1	@title	dissertacao	relacionamento
article/year	1	@year	dissertacao	relacionamento
person/name	1	name	dissertacao	relacionamento
article/{author}person	1	author	dissertacao	relacionamento
article/{reviewer}person	1	reviewer	dissertacao	relacionamento
person/{author}article	1	[local_name. = "author"]/..	dissertacao	relacionamento
person/{reviewer}article	1	[local_name(.) = "reviewer"]/..	dissertacao	relacionamento
institution/person	1	../..	dissertacao	relacionamento



Conceito/Relacionamento	*Nº da Fonte*	*Mapeamento*
Article ← {author} Person	1	[local_name(.) = "author"]/..
Article ← {author} Person	2	article

FIGURA 4.3 – Exemplo 2 de mapeamento de um relacionamento

4.2 Tradução de CXPath para XPath 1.0

Esta seção apresenta o processo da tradução de expressões CXPath para XPath 1.0 de uma determinada fonte XML. O mecanismo da tradução consiste em reescrever a expressão segundo a informação de mapeamento armazenada para esta fonte. Cada referência a um conceito assim como cada relacionamento entre os conceitos encontrados na expressão CXPath são substituídos por sua informação de mapeamento correspondente na fonte de XML.

O processo da tradução de uma expressão CXPath para XPath 1.0 segue basicamente as seguintes regras:

1. O mecanismo de tradução de CXPath analisa um a um os conceitos da expressão CXPath da esquerda para a direita;

2. Cada construção "/", "["ou "]"encontrada na entrada é escrita na saída com exceção a "/"inicial de expressões absolutas;

3. Ao analisar um conceito *C*, se este conceito for o primeiro conceito de uma expressão absoluta CXPath do tipo */C*, é escrita na saída a informação de mapeamento associada a este conceito. Neste caso, o operador da navegação (/) que precede o conceito não é escrito à saída (conforme a regra 2);

4. Quando um conceito Cj analisado na entrada for parte de uma expressão de caminho relativa CXPath do tipo $Ci/\{R.P\}Cj$ ou um filtro $Ci[\{R.P\}Cj...]$ da navegação do predicado, o mapeamento do relacionamento R de Ci para o papel P de Cj é escrito na saída. A construção opcional de CXPath {R}, {R.P},

{P}, quando encontrada, mesmo não sendo escrita na saída, é necessária e é considerada para a correta identificação do mapeamento do relacionamento.

Um conceito faz parte de uma expressão absoluta quando não há contexto anterior ao conceito em questão. Isto ocorre nos casos em que ele é o primeiro conceito de uma expressão absoluta. Os próximos conceitos farão parte de expressões CXPath relativas relacionadas com o nodo contexto imediatamente anterior. Essa diferenciação é importante para selecionar a escolha correta entre mapeamento de conceitos e relacionamentos, conforme as regras 3 e 4.

5. O predicado de seleção <operador de seleção> <valor> encontrado na entrada é escrito na saída (= *"José da Silva"*, porque exemplo). Se o valor do predicado de seleção corresponder a uma outra expressão CXPath, o processo de tradução é chamado recursivamente entrando em quantos níveis forem necessários e depois retornando a medida que estas *sub-expressões* CXPath recursivas vão sendo traduzidas e encerradas;

6. Em alguns mapeamentos de relacionamentos envolvendo predicados, pode ocorrer uma escrita de dois operadores de navegação: "/["; para prover a tradução correta, definiu-se que o operador "["tem precedência sobre o operador "/", uma vez que "["é necessário para expressar os predicados desejados. Assim, quando a seqüencia "/["é encontrada na saída ela é substituída por "[". Isto será apresentado no exemplo 4.6 onde pode ser observado que não foi reescrito na saída o operador de navegação *"/"* entre o mapeamento do conceito *person* e o mapeamento do relacionamento de *person - article*.

As expressões CXPath sempre são construídas a partir das informações do modelo conceitual. As estruturas abstratas *relacionamento R* e *papel P* podem ou não estar presentes, tudo dependendo de como a navegação entre dois conceitos estiver representada no modelo conceitual. Se estas estruturas estiverem presentes, a informação de mapeamento deve ter visões XPath para cada papel e relacionamento possível.

Outro aspecto importante é que nem sempre a expressão imediatamente anterior ao operador de navegação é o *nodo de contexto* de uma navegação para conceito posterior. Em outras palavras, ao analisar o conceito Cj de uma expressão relativa CXPath do tipo Ci[predicado]/Cj ou Ci[predicate][Cj ...], deve-se mapear o relacionamento de Ci para Cj. Dessa forma, deve-se guardar a informação de contexto antes da análise do predicado. Após a análise do predicado deve-se retornar a esse contexto anterior para analisar o relacionamento correto.

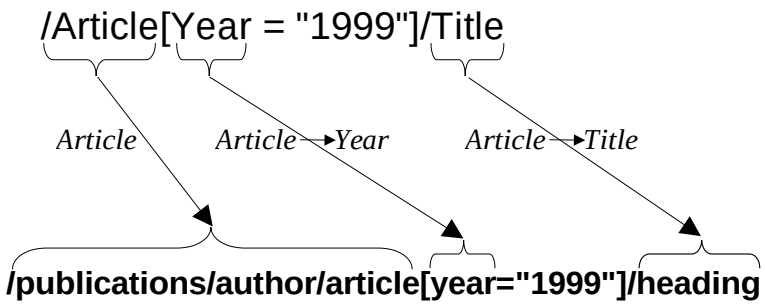
Como o nodo raiz pode ser diferente em cada fonte XML, o modelo conceitual é abstrato e não possui um *ponto de entrada*. O usuário da consulta pode então escolher qualquer ponto de entrada para suas expressões. É por esta razão que o processo de tradução primeiramente mapeia um conceito, pois precisa traçar toda navegação necessária na fonte para atingir o ponto de entrada escolhido pelo construtor da consulta.

Após isso, a tradução prossegue mapeando os relacionamentos entre os conceitos até atingir o conceito léxico "folha"desejado como resultado.

No caso do operador de seleção conter outras expressões absolutas aninhadas, o mecanismo de tradução analisa recursivamente em um novo escopo cada uma das expressões, retornando ao escopo original ao concluir essas traduções parciais.

4.2.1 Exemplo 1

Na figura 4.4 é mostrado um exemplo da tradução da consulta CXPath que busca *títulos dos artigos escritos em 1999*.



Conceito/Relacionamento	*Nº da Fonte*	*Mapeamento*
article	2	/publications/author/article
article → year	2	year
article → title	2	heading

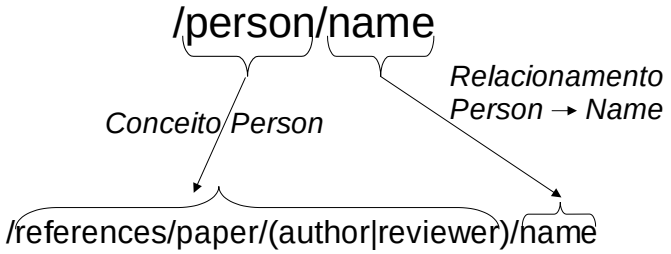
FIGURA 4.4 – Tradução de uma consulta CXPath para XPath (fonte 2)

A seguir são apresentados os passos seguidos pelo mecanismo de tradução neste exemplo, conforme as regras apresentadas neste capítulo.

- O primeiro operador de navegação "/"não é escrito na saída.
- O conceito *Article* é o primeiro conceito da expressão absoluta e assim, o respectivo mapeamento para este conceito é escrito na saída. O nodo contexto é *Article*.
- O operador "["é escrito na saída
- O conceito *year* encontrado faz parte de numa expressão relativa e assim o mapeamento da navegação pelo relacionamento de Article para year é escrito na saída. O nodo contexto passa a ser *year*.
- Como o predicado de seleção *="1999"* não possui outras expressões CXPath ele é escrito à saída.
- O operador "]"é escrito na saída. O nodo contexto volta a ser *Article*.
- Encontrado conceito *title*. O conceito *title* encontrado faz parte de numa expressão relativa. O mapeamento da navegação pelo relacionamento de *Article* para *title* é escrito na saída. O nodo contexto passa a ser *title*.
- A expressão CXPath chegou ao fim. A tradução é concluída.

4.2.2 Exemplo 2

No exemplo da figura 4.5, a consulta é traduzida para fonte 1. Na expressão CXPath é solicitado o "nome de todas as pessoas". A consulta traduzida precisa então percorrer dois caminhos diferentes na fonte XML para alcançar os elementos *autor* e *reviewer* que representam todas as instâncias de pessoas nesta fonte. O mapeamento de conceito utiliza uma *visão XPath 1.0* como segue, */references/paper/(author|reviewer)*, e assim consegue navegar pelos dois caminhos que devem ser percorridos. Isso demonstra o poder do mecanismo de mapeamento através de visões, pois basta o correto armazenamento das visões para que se obtenha sucesso na tradução.



Conceito/Relacionamento	*Nº da Fonte*	*Mapeamento*
person	1	/references/paper/(author|reviewer)
person → name	1	name

FIGURA 4.5 – Tradução de uma consulta CXPath para XPath (fonte 1)

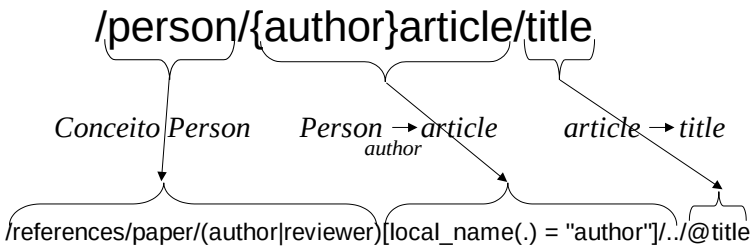
A fonte 2 não possui informação sobre revisores. Se esta mesma consulta CXPath for traduzida para fonte 2 serão retornados apenas os autores, uma vez que eles representam todo o conjunto de pessoas existentes na fonte. Nesse caso o mapeamento do conceito *person* será simplesmente */references/paper/author* e a tradução da consulta CXPath */person/name* para XPath será */references/paper/author/name*.

4.2.3 Exemplo 3

Neste exemplo é apresentado um mapeamento mais complexo, que é capaz de adequar a direção da navegação feita pelo usuário à direção de navegação possível, de acordo com a estrutura hierárquica da fonte XML. Como o usuário pode iniciar a consulta CXPath em qualquer conceito do modelo conceitual e navegar nos relacionamentos em qualquer direção, as visões XPath 1.0 presentes na informação de mapeamento tem a capacidade de adaptar a navegação criada pelo usuário as peculiaridades das diferentes representações da informação nas fontes.

Na figura 4.6 é apresentado o exemplo que solicita os títulos de artigos escritos pelos autores. Primeiramente é traduzido o conceito *person*. Após o conceito *person*, a expressão CXPath navega para o conceito *article* pelo relacionamento *author*. Para representar essa navegação na fonte 1 é necessário primeiramente fixar o contexto local apenas nas instâncias de autor, uma vez que o usuário solicitou o relacionamento de autoria entre *person* e *article*. Após isso, deve-se subir no relacionamento

hierárquico da fonte com o operador *"..."*, a fim de alcançar o contexto do conceito *article* representado na fonte pelo elemento XML *paper*. O último mapeamento representa o acesso ao contexto do conceito léxico *title* que na fonte é um atributo do elemento XML *paper*.



Conceito/Relacionamento	*Nº da Fonte*	*Mapeamento*
person	1	/references/paper/(author|reviewer)
person →(author) article	1	[local_name(.) = "author"]/..
article → title	1	@title

FIGURA 4.6 – Tradução de uma consulta CXPath para XPath (fonte 1)

A consulta traduzida resultante é perfeitamente funcional e correta. Pode-se questionar entretanto, porque a consulta não é traduzida para algo mais conciso, como */references/paper/author/../@title*. Para isso é necessário construir mecanismos adicionais para detectar e desconsiderar caminhos que não existam nas fontes e analisar a possibilidade de sua exclusão. Esse trabalho adicional é interessante, mas não necessário para atingir os objetivos propostos, e por isso foi deixado fora do escopo dessa dissertação.

4.3 Tradução de CXQuery para XQuery

Esta seção apresenta o processo de tradução das consultas CXQuery para XQuery 1.0 de uma determinada fonte XML. Este mecanismo também faz a validação da consulta segundo a gramática CXQuery. Seu funcionamento consiste em reescrever a consulta substituindo as expressões CXPath encontradas entre os operadores CXQuery, pela tradução XPath 1.0 equivalente, fornecida pelo mecanismo de tradução CXPath - XPath 1.0. Sendo assim, o mecanismo de tradução de CXQuery para XQuery utiliza o mecanismo de tradução apresentado no capítulo anterior, organizando a tarefa de tradução. O processo da tradução de uma consulta CXQuery para XQuery 1.0 segue basicamente as seguintes regras:

1. Os construtores de elementos XML normalmente utilizados na formatação do documento de resultado da consulta não são modificados e simplesmente são reescritos pelo mecanismo de tradução de CXQuery - XQuery 1.0;

2. Nesse caso o corpo da consulta CXQuery só é avaliado dentro dos delimitadores de início e fim de consulta: "{"e "}". Se a consulta CXQuery for declarada sem o uso de construtores XML, estes delimitadores não são necessários e a

validação e tradução da consulta é iniciada imediatamente no primeiro caracter encontrado;

3. Nas construções *"for <variável de iteração > in <contexto>"* do corpo da consulta CXQuery, o operador *"for"*, o nome da variável de iteração e o operador *in* são simplesmente reescritos na saída.

   Já o contexto da variável de iteração é traduzido e escrito na saída após o operador *"in"*.

   Além disso, para fins de uso posterior, o nome da variável é armazenado temporariamente juntamente com o contexto desta variável. Estas informações são necessárias na tradução de expressões CXPath que utilizem as variáveis de iteração.

   Da mesma forma que XQuery 1.0, nestas expressões partes das expressões de caminho são compostas pelas variáveis de iteração que fazem a varredura das instâncias dos conceitos;

4. As expressões CXPath de iteração presentes dentro de uma consulta CXQuery, são traduzidas utilizando informações armazenadas na tradução dos comandos *"for"*. Primeiramente o nome da variável da expressão é substituído pelo respectivo contexto CXPath armazenado para essa variável. Com isso obtém-se uma expressão absoluta CXPath que é traduzida utilizando o mecanismo de tradução apresentado no capítulo anterior.

   Deste modo, de posse da expressão absoluta traduzida para XPath 1.0 deve-se recolocar a variável de iteração na expressão traduzida.

   Para fazer essa operação, o mecanismo busca o contexto armazenado, traduz ele para XPath, e localiza o contexto na expressão absoluta substituindo-o pelo nome da variável de iteração.

   Assim obtém-se uma expressão com a variável de iteração, seguida da expressão XPath 1.0 correspondente. Esse mecanismo pode ser melhor compreendido no exemplo de tradução CXQuery - XQuery 1.0 deste capítulo.

5. Nas construções *"where <expressão CXPath de iteração> <operador de comparação> <critério de comparação>"*, são reescritos na saída o operador *"where"* bem como o operador de comparação e o critério. Já a expressão CXPath de iteração sofre um complexo processo de tradução conforme mostrado no item 4;

6. Nas construções *"... return <construções XML> <expressão CXPath de iteração> <construções XML>"* são reescritos na saída o operador *"return"* bem como todas as construções XML presentes no contexto da cláusula return. A expressão CXPath de iteração é traduzida conforme apresentado no item 4;

7. Após o encerramento do corpo da consulta com o operador *"}"*, se houverem construtores de elementos XML eles são reescritos sem modificações;

8. No caso do corpo da consulta CXQuery possuir apenas uma expressão CXPath absoluta, o mecanismo de tradução de CXQuery apenas aciona o mecanismo de tradução CXPath retornando o resultado obtido para saída após a tradução;

4.3.1 Exemplo 1

A seguir é apresentado um exemplo de tradução de consulta CXQuery para XQuery 1.0 da W3C. Este exemplo aplica a tradução para fonte 1 da consulta "autores do artigo X e suas respectivas instituições". A saída é um documento XML cuja raiz é o elemento *autores*. Este elemento será composto de uma lista de elementos autor e instituição.

Uma vez que a tradução das expressões CXPath já foi apresentada no capítulo 4.2, aqui serão exemplificadas apenas as questões relevantes aos principais passos da tradução de CXQuery para XQuery 1.0. A consulta CXQuery de entrada é apresentada abaixo. Após isso, é feita uma explicação de cada linha da consulta XQuery 1.0 que vai sendo escrita na saída.

Consulta CXQuery

```
<Autores>
{
for $pe in /person
where $pe/{author}article/title = "X"
return
<autor>
$pe/name
</autor>
<instituição>
$pe/institution
</instituição>
}
</Autores>
```

1. Os construtores iniciais de elementos XML são reescritos:

   ```
   <Autores>
   ```

2. Após a análise dos construtores iniciais é encontrado um delimitador de início de consulta: "{". Na seqüência, o operador *"for"* e o nome da variável de iteração são reescritos na saída. O nome da variável é armazenado juntamente com o contexto desta variável que aparece após o operador *in*. Esse contexto é traduzido para XPath 1.0 pelo mecanismo de tradução CXPath apresentado no capítulo 4.2. Esta tradução é escrita na saída após o operador *"in"*.

   ```
   {
   for $pe in /references/paper/(author|reviewer)
   ```

   variáveis e contextos armazenados:

   - contexto de $pe : /person

3. São reescritos na saída o operador *"where"* bem como o operador de comparação e o critério. Para traduzir a expressão CXPath $pe/{author}article/title primeiramente substitui-se o nome da variável da expressão pelo respectivo contexto CXPath armazenado para essa variável. Após a tradução a variável de iteração é substituída pela tradução de seu contexto para XPath.

   Tradução da expressão CXPath com variável de iteração:

```
CXPath:  $pe/{author}article/title

Substituição do identificador da variável de iteração pelo
contexto armazenado: "/person"

CXPath:  /person/{author}article/title

Tradução XPath 1.0 :
/references/paper/(author|reviewer)[local_name(.) =
"author"]/../@title

Tradução do contexto de $pe :
"/references/paper/(author|reviewer)"

Substituição dessa sub expressão pelo identificador da variável.
$pe/[local_name(.) = "author"]/../@title
```

   Cláusula where traduzida:

```
where $pe/[local_name(.) = "author"]/../@title = "X"
```

4. O operador *"return"* é encontrado e escrito na saída. As construções de elementos XML presentes no contexto da cláusula return também são escritos na saída. As expressões CXPath de iteração são traduzidas conforme apresentado no item anterior;

   Tradução da primeira expressão CXPath da cláusula return com variável de iteração:

```
$pe/name

Substituição do identificador da variável de iteração pelo
contexto armazenado: "/person"

CXPath:  /person/name
```

```
Tradução XPath 1.0:  /references/paper/(author|reviewer)/name

Tradução do contexto de $pe :
"/references/paper/(author|reviewer)"

Substituição dessa sub expressão pelo identificador da variável:
$pe/name
```

Tradução da segunda expressão CXPath com variável de iteração:

```
$pe/institution

Substituição do identificador da variável de iteração pelo
contexto armazenado: "/person"

CXPath:  /person/institution

Tradução XPath 1.0 :
/references/paper/(author|reviewer)/institution/@description

Tradução do contexto de $pe :
"/references/paper/(author|reviewer)"

Substituição dessa sub expressão pelo identificador da variável:
$pe/institution/@description
```

Cláusula return traduzida:

```
return
<autor>
$pe/name
</autor>
<instituição>
$pe/institution/@description
</instituição>
```

5. Encontrados *"}"*, os construtores de elementos XML externos finais são reescritos sem modificações;

```
}
</Autores>
```

Consulta Traduzida para XQuery 1.0 da fonte 1

```
<Autores>
{
for $pe in /references/paper/(author|reviewer)
where $pe/[local_name(.) = "author"]/../@title = "X"
return
<autor>
$pe/name
</autor>
<instituição>
$pe/institution/@description
</instituição>
}
</Autores>
```

O mecanismo de tradução apresentado neste capítulo está implementado segundo a arquitetura apresentada na figura 4.7.

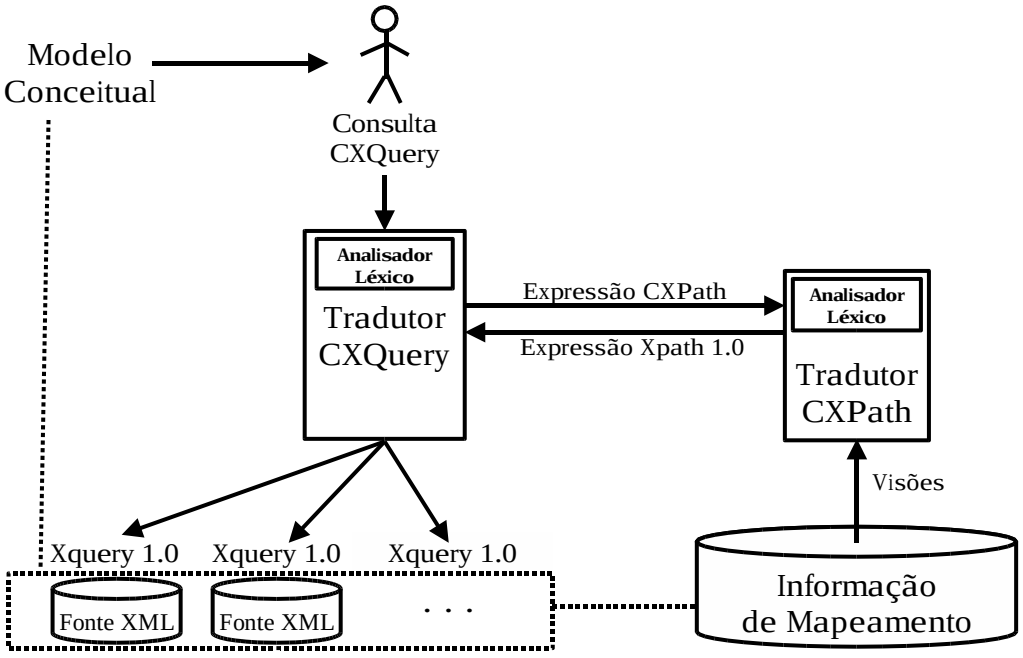


FIGURA 4.7 – Arquitetura do mecanismo de tradução

O processo de tradução se inicia quando o usuário cria uma consulta CXQuery com base no modelo conceitual para efetuar uma consulta sobre as fontes XML. O mecanismo de tradução CXQuery inicia o processo de tradução interagindo com o tradutor CXPath. A medida que expressões CXPath são encontradas elas são enviadas ao tradutor CXPath que acessa a informação de mapeamento e devolve a expressão traduzida em XPath 1.0. O tradutor CXQuery não acessa diretamente a

informação de mapeamento, ele é responsável pelo controle e armazenamento dos contextos das variáveis de iteração. Também é responsável pela análise léxica da consulta CXQuery.

A linha pontilhada representa a interação com as arquiteturas fora do escopo deste trabalho. Estas, são responsáveis pela criação do modelo conceitual a partir da integração das fontes XML, e pelo armazenamento da informação de mapeamento entre o paradigma XML e o paradigma conceitual.

Após concluída a tradução, o tradutor devolve a(s) consulta(s) traduzida(s) na linguagem XQuery 1.0 para que se promova a efetiva consulta nas fontes XML desejadas.

Os dados retornados por essas consultas são independentes entre si. A integração destes resultados para apresentação uniformizada ao usuário, lidando e definindo ações em caso de ocorrências de lacunas na representação de determinadas informações, não pertencem ao escopo do presente trabalho. Estas lacunas podem ocorrer devido a heterogeneidade das formas de representação da informação. Também não são tratados problemas decorrentes da decomposição de consultas. Entre esses problemas, podem ser citados: a busca de informação fragmentada em várias fontes, união de informação de fontes e interações das informações das fontes entre si. O enfoque no presente trabalho é de concentrar esforços no problema de tradução de consultas. Dessa forma, são apresentados mecanismos para traduzir individualmente consultas sobre o modelo conceitual para consultas em linguagens XML.

5 Conclusão

Este trabalho propõe uma adaptação da linguagem XQuery 1.0 da W3C chamada linguagem de consulta CXQuery (Conceptual XQuery). Essa linguagem possibilita a construção de consultas sobre um modelo conceitual global permitindo que o usuário possa consultar um conjunto de fontes XML heterogêneas de maneira integrada. Essa linguagem de consulta é definida com base em estudos de várias linguagens existentes. Estes estudos são guiados por critérios pré-definidos no capítulo 2.2. Além disso são formalizados quais são e como são as construções e operadores da linguagem CXQuery através da criação da gramática de CXQuery apresentada no anexo 5,.

Também é definido e desenvolvido um mecanismo de mapeamento que associa os conceitos e relacionamentos do modelo conceitual a visões XPath 1.0 em termos da estrutura, dos elementos e atributos das fontes heterogêneas. Esse mapeamento permite a tradução das expressões de caminho CXPath, possibilitando assim, também a tradução das consultas CXQuery para a linguagem XQuery 1.0 da W3C. Tanto na tradução das expressões CXPath para XPath 1.0, quanto na tradução de CXQuery para XQuery 1.0 são definidas as regras e procedimentos utilizados pelo mecanismo de tradução. Após a finalização da tradução, a consulta XQuery 1.0 gerada permite o acesso aos dados de uma fonte XML qualquer.

Dessa forma as contribuições do presente trabalho são:

- a linguagem de consulta CXQuery e a linguagem de expressões CXPath para consultas sobre um modelo conceitual de integração de fontes XML heterogêneas.
- a definição da metodologia de representação da informação de mapeamento do modelo conceitual para uma fonte XML qualquer através de visões XPath 1.0.
- o mecanismo de tradução das expressões CXPath para XPath 1.0 e das consultas CXQuery para XQuery 1.0.

Uma característica do escopo definido para este trabalho é a de que as consultas são traduzidas para uma fonte qualquer independentemente da informação das outras fontes. Em fontes que possuem lacunas de representação ou informação distribuída, pode ser necessário interagir com informações de outras fontes para possibilitar a tradução.

Sendo assim, pode ser citado como trabalho futuro a tarefa de tratar o problema da decomposição das consultas, que não é abordado no presente trabalho. Essa tarefa envolve problemas como a busca de informação fragmentada e distribuída em várias fontes, união da informação de fontes através de junções e outros casos em que as interações das informações das fontes entre si se mostrem necessárias. Também irá lidar com problemas de otimização de consultas, eliminação de redundâncias e pré-processamento, eliminando acessos desnecessários as fontes.

Como conseqüência da tradução independente das consultas, os resultados obtidos nessas consultas, também são instâncias independentes.

Outro trabalho futuro pode construir um mecanismo de integração de resultados, e assim fornecer uma apresentação uniformizada e integrada dos resultados ao

usuário. Esse futuro trabalho deve lidar com o problema de, por exemplo, apresentar técnicas que possam definir como apresentar um resultado integrado em caso de ocorrências de lacunas na representação de determinadas informações das fontes.

Outra questão diz respeito a exigência de retorno léxico das consultas, sendo proibido consultas que retornem conceitos compostos por outros conceitos *(conceitos não-léxicos)*. Juntamente com a integração dos resultados proposta no parágrafo anterior, pode ser citado como trabalho futuro a avaliação da utilidade e a possibilidade de se integrar instâncias de resultados não-léxicos e assim permitir consultas CXQuery que retornem ambos os conceitos.

Dessa forma foram apresentadas as contribuições científicas do presente trabalho e tópicos inexplorados que podem ser estudados em futuros trabalhos.

Em anexo, por fim, são apresentadas a gramática da linguagem CXQuery, uma breve descrição do protótipo desenvolvido e os autômatos da análise léxica deste protótipo.

Anexo 1 A gramática de CXQuery

Introdução

Uma gramática é um mecanismo para gerar as sentenças ou palavras de uma linguagem [PRI 2001] e é definida pela quádrupla (N,T,P,S) onde:

- N é o conjunto de símbolos não-terminais (variáveis)
- T é o conjunto de símbolos terminais (constantes)
- P é o conjunto de regras de produção
- S é o símbolo inicial da gramática

A definição formal da gramática da linguagem CXQuery é em linhas gerais uma modificação e simplificação de um subconjunto da gramática de XQuery publicada em [W3C 2003] e [DRA 2002]. Como uma consulta CXQuery utiliza expressões CXPath, dentro da gramática de CXQuery há regras de produção que validam também as expressões CXPath. A gramática CXQuery é apresentada abaixo:

Gramática CXQuery => (N,T,P,S)

N - Conjunto de Símbolos não terminais da gramática CXQuery

N = CXQuery, FWRExpr, ForClause, WhereClause, QuantifiedExpr, RangeExpr, Constructor, ElementConstructor, ElementContent, EnclosedExpr, CXPathExpr, RelativePathExpr, StepExpr, ForwardStep, PrimaryExpr, Predicates, Literal, NumericLiteral, IntegerLiteral, DecimalLiteral, DoubleLiteral, relationship, role, NodeTest, VarName, QName, NCNameChar, Digits, Letter, GeneralComp, StringLiteral

T - Conjunto de Símbolos terminais da gramática CXQuery

```
T = {[a-z], [A-Z], [0-9], "for" , "in" , "where", "return", "=", "!=",
"<", "<=", ">", ">=", "<", "/>", ">", "</", "/", "}", "}",
 "$", "[", "]", ".", "e", "E", "+", "-", "_"}
```

P - Conjunto de regras de produção da gramática CXQuery

```
CXQuery    ::=    FWRExpr?
```

```
FWRExpr ::= (ForClause+ WhereClause? "return")? QuantifiedExpr

ForClause    ::=    <"for" "$"> VarName "in" CXPathExpr (<"for"
"$"> VarName "in" CXPathExpr)*

WhereClause  ::=    "where" QuantifiedExpr

QuantifiedExpr   ::= (RangeExpr) (GeneralComp RangeExpr)?

RangeExpr ::= (Constructor |  CXPathExpr)

GeneralComp    ::=    "=" |  "!=" |  "<" |  "<=" |  ">" |  ">="

Constructor    ::=    ElementConstructor

ElementConstructor ::= "<"QName("/>"|(">"ElementContent*
"</"QName">"))

ElementContent ::=
NCNameChar|ElementConstructor|CXPathExpr|EnclosedExpr
```

Quando a consulta possui construtores XML, A produção a seguir faz a análise do corpo da consulta CXQuery entre "{ }"

```
EnclosedExpr    ::=    "{" FWRExpr "}"
```

A próxima regra de produção e suas derivações efetuam a análise das expressões CXPath dentro de uma consulta CXQuery

```
CXPathExpr    ::= ("/" RelativePathExpr?) | RelativePathExpr

RelativePathExpr    ::=    StepExpr ("/" StepExpr)*

StepExpr ::=  (ForwardStep | PrimaryExpr) Predicates

ForwardStep ::= ("{" relationship ( ("}") | ("." role "}") )?
NodeTest

PrimaryExpr    ::=    Literal | ("$" VarName)

Predicates    ::=    "[" (CXPathExpr) (GeneralComp literal)? "]"

Literal    ::=    NumericLiteral |  StringLiteral

NumericLiteral ::= IntegerLiteral | DecimalLiteral | DoubleLiteral

StringLiteral ::=    ('"' (('"' '"') | [^"])* '"') | ("'" (("'"
"'") | [^'])* "'")
```

```
IntegerLiteral    ::=    Digits

DecimalLiteral    ::=    ("." Digits) |  (Digits "." [0-9]*)

DoubleLiteral ::= (("." Digits) | (Digits("."[0-9]*)?))("e"| "E")
("+" | "-")? Digits

Digits    ::=    [0-9]+

NodeTest    ::=    QName

relationship    ::=    QName

role    ::=    QName

VarName    ::=    QName

QName ::= (Letter) (NCNameChar)*

NCNameChar ::= Letter | Digit | "_"

Letter    ::=    [a-z|A-Z]+
```

S - Símbolo inicial da gramática CXQuery

S = CXQuery

Anexo 2 Protótipo desenvolvido

Neste capítulo será apresentado o funcionamento básico do protótipo de tradução CXQuery-XQuery desenvolvido para Web.

A informação de mapeamento que possui as visões XPath para os conceitos e relacionamentos foi armazenada num banco de dados relacional.

Algumas considerações sobre a simplificação do protótipo:

Não foram desenvolvidas rotinas para impedir a consulta solicitando retorno não léxico. Também não é feita a validação dos construtores de elementos XML que podem ser utilizados nas consultas CXQuery.

Além disso, até o alcance dos testes efetuados, toda consulta válida em CXQuery é corretamente identificada e traduzida. Entretanto, determinados erros sintáticos específicos perante a sintaxe CXQuery não são rejeitadas pelo analisador léxico do protótipo de tradução.

Esses defeitos ocorrem pois o autômato e o mecanismo de análise léxica não eram os objetivos centrais da criação do protótipo. A criação do protótipo foi feita apenas como uma maneira de testar pragmaticamente os conceitos aqui defendidos *para as consultas CXQuery válidas*, no que tange ao mapeamento e o mecanismo de tradução, validando seu funcionamento. Mesmo já tendo claramente definido as características de CXQuery antes da construção do protótipo, a linguagem CXQuery só foi definida formalmente através de sua gramática após a construção do protótipo de tradução. Como trabalho futuro, pode ser gerado automaticamente, a partir de geradores léxicos e sintáticos, um analisador completo da linguagem CXQuery em total conformidade com a gramática.

A página inicial do protótipo pode ser vista na figura B.1.

Na área de texto é colocada a consulta CXQuery. Como muitos modelos conceituais representando suas respectivas fontes XML podem ser armazenados na informação de mapeamento, deve-se informar o nome do modelo conceitual desejado. Escolhe-se para quais fontes deseja-se a tradução e clica-se em *translate to Xquery*. A opção *Show Analisys* é útil para depurar passo a passo o processo de tradução mas pode tornar bastante lento o processo dependendo do tamanho da consulta.

Na página inicial ainda há um link de ajuda com uma breve informação de auxílio geral e exemplos de consultas que podem ser utilizadas para verificar o funcionamento da tradução. Também há um link para uma interface que acessa a informação de mapeamento armazenada e permite apagar, inserir ou simplesmente visualizar os mapeamentos armazenados.

Após efetuada a tradução, a página contendo a consulta CXQuery inicial e as traduções efetuadas pelo protótipo para a linguagem XQuery 1.0 nas fonte 1 e 2 (figuras 3.1 e 3.1) pode ser observada na figura B.2

Entrando no link *Mapping Information* e depois em *Access the Mapping Information* é mostrada a página onde podem ser vistas as visões XPath que mapeiam os conceitos e relacionamentos dos modelos conceituais para cada uma das suas respectivas fontes. A página com essa informação de mapeamento pode ser vista na figura B.3

Cada registro contém uma visão que fornece o mapeamento de um conceito ou relacionamento. Campos:

- *ID XML Source:* Fonte do mapeamento

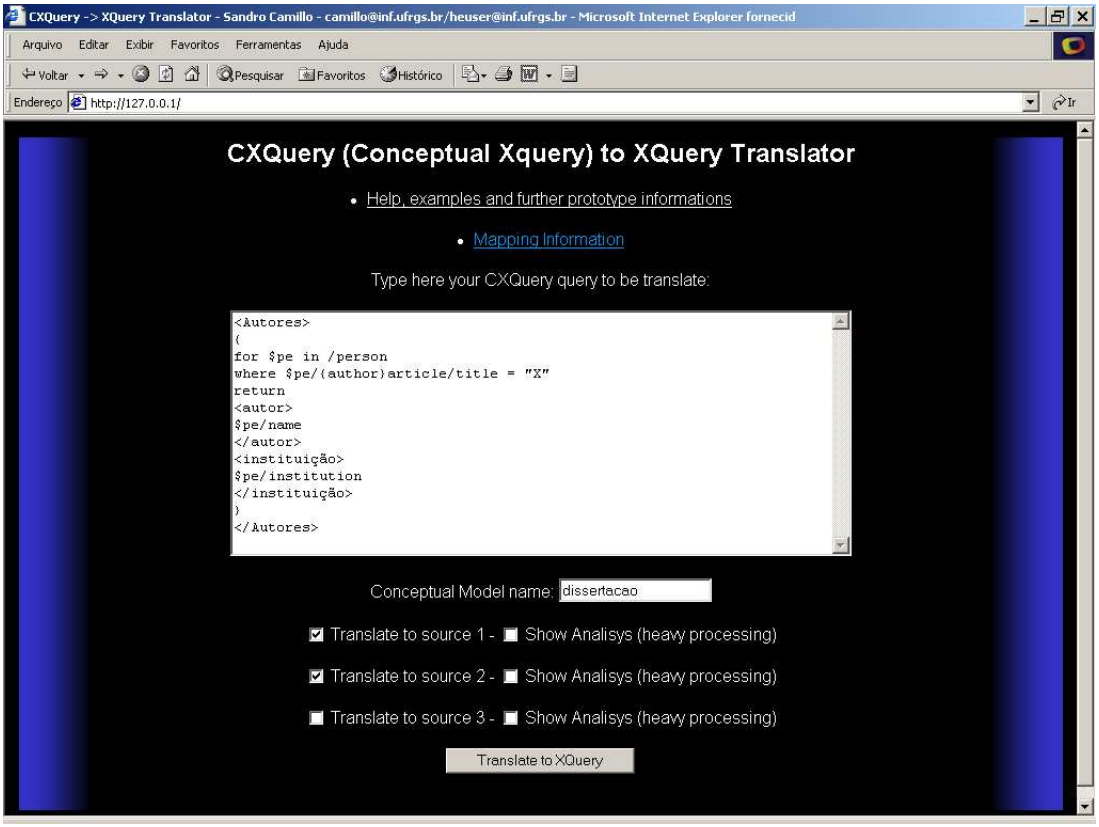


FIGURA B.1 – Tela inicial do protótipo

- *Concept/relationship:* Nome do conceito ou do relacionamento do modelo conceitual a ser mapeado
- *Mapping:* a respectiva visão XPath do conceito ou relacionamento nessa fonte (mapeamento propriamente dito)
- *ID Model:* O nome do modelo conceitual

A página que permite inserção de novos mapeamentos pode ser vista na figura B.4.

No campo *ID XML Source* deve ser informado qual é a fonte do mapeamento a ser inserido (número 1, 2 ou 3). No campo *Concept/relationship* se informa o nome do conceito ou do relacionamento do modelo conceitual e no campo *Mapping* a sua respectiva visão que o representa nessa fonte. No campo *ID Model* deve ser informado o nome do modelo conceitual e no campo *Mapping kind* se o mapeamento é de conceito ou relacionamento.

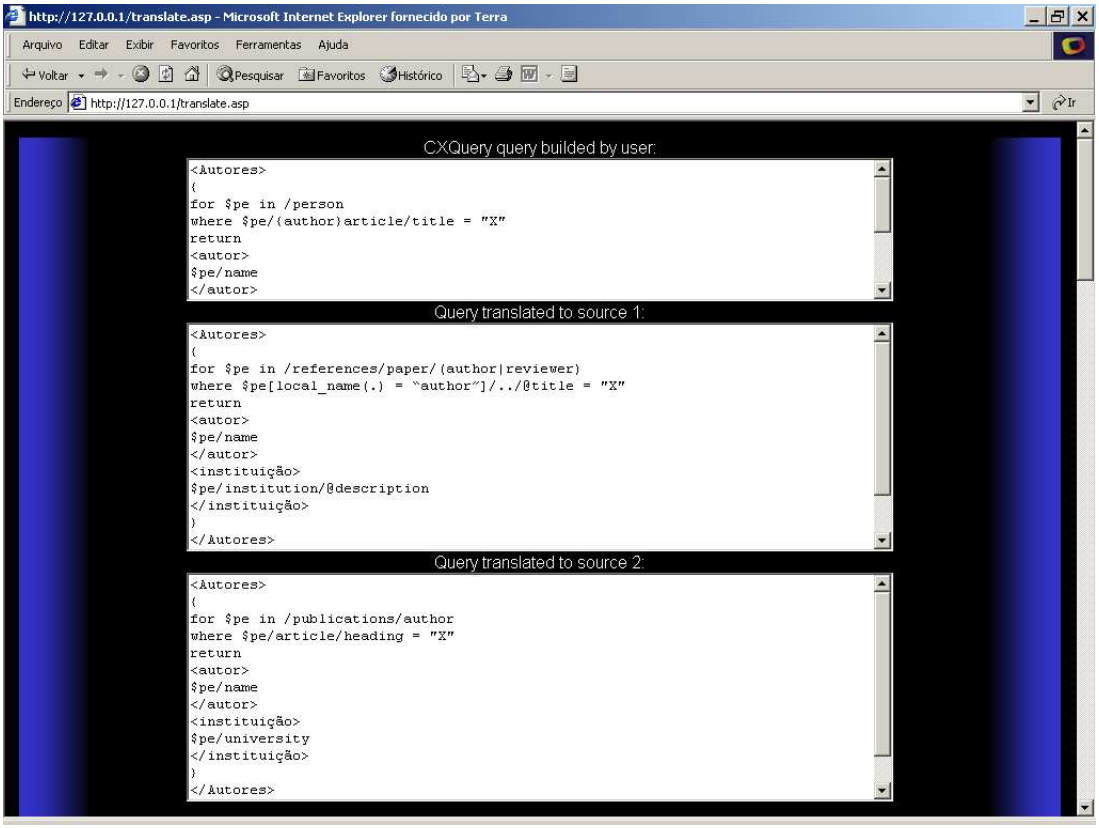
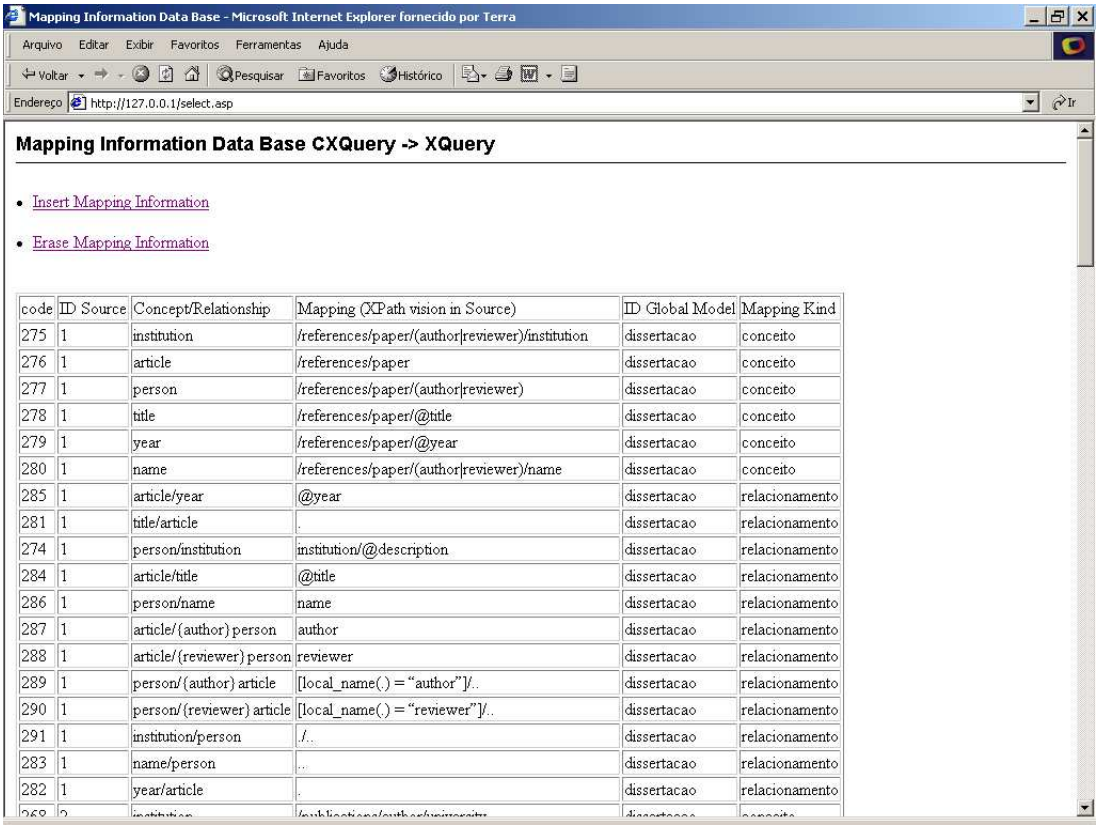


FIGURA B.2 – Página com as traduções para XQuery das fontes



code	ID Source	Concept/Relationship	Mapping (XPath vision in Source)	ID Global Model	Mapping Kind
275	1	institution	/references/paper/(author|reviewer)/institution	dissertacao	conceito
276	1	article	/references/paper	dissertacao	conceito
277	1	person	/references/paper/(author|reviewer)	dissertacao	conceito
278	1	title	/references/paper/@title	dissertacao	conceito
279	1	year	/references/paper/@year	dissertacao	conceito
280	1	name	/references/paper/(author|reviewer)/name	dissertacao	conceito
285	1	article/year	@year	dissertacao	relacionamento
281	1	title/article	.	dissertacao	relacionamento
274	1	person/institution	institution/@description	dissertacao	relacionamento
284	1	article/title	@title	dissertacao	relacionamento
286	1	person/name	name	dissertacao	relacionamento
287	1	article/{author} person	author	dissertacao	relacionamento
288	1	article/{reviewer} person	reviewer	dissertacao	relacionamento
289	1	person/{author} article	[local_name(.) = "author"]/..	dissertacao	relacionamento
290	1	person/{reviewer} article	[local_name(.) = "reviewer"]/..	dissertacao	relacionamento
291	1	institution/person	/..	dissertacao	relacionamento
283	1	name/person	..	dissertacao	relacionamento
282	1	year/article	.	dissertacao	relacionamento

FIGURA B.3 – Página de visualização da informação de mapeamento

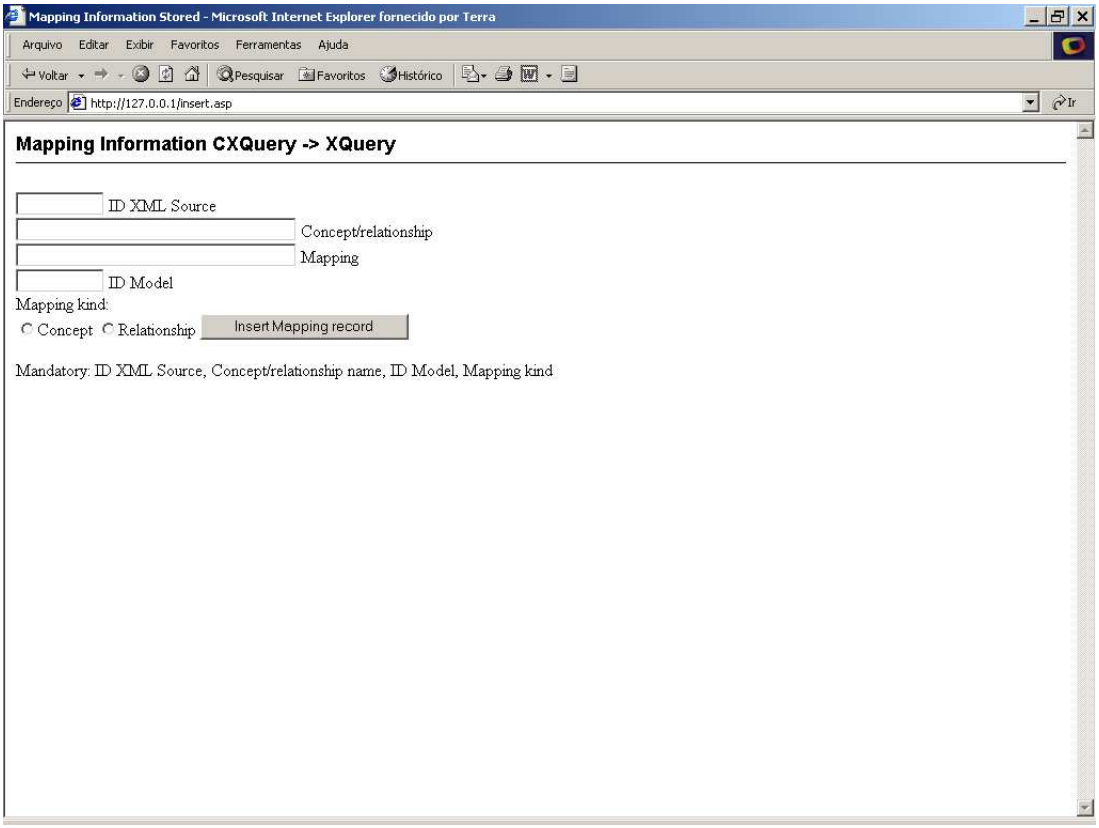


FIGURA B.4 – Tela de inserção de novos mapeamentos

Anexo 3 Autômato da Análise Léxica e tradução

Nesta seção serão apresentados os autômatos desenvolvidos para dar suporte a implementação de um protótipo para analisar e traduzir as consultas CXQuery. As expressões CXPath encontradas na consulta CXQuery são analisadas e traduzidas por um modulo de software independente e possui seu próprio autômato. Na figura C.1 é apresentado o autômato para CXPath e na figura C.2 é apresentado o autômato para CXQuery.

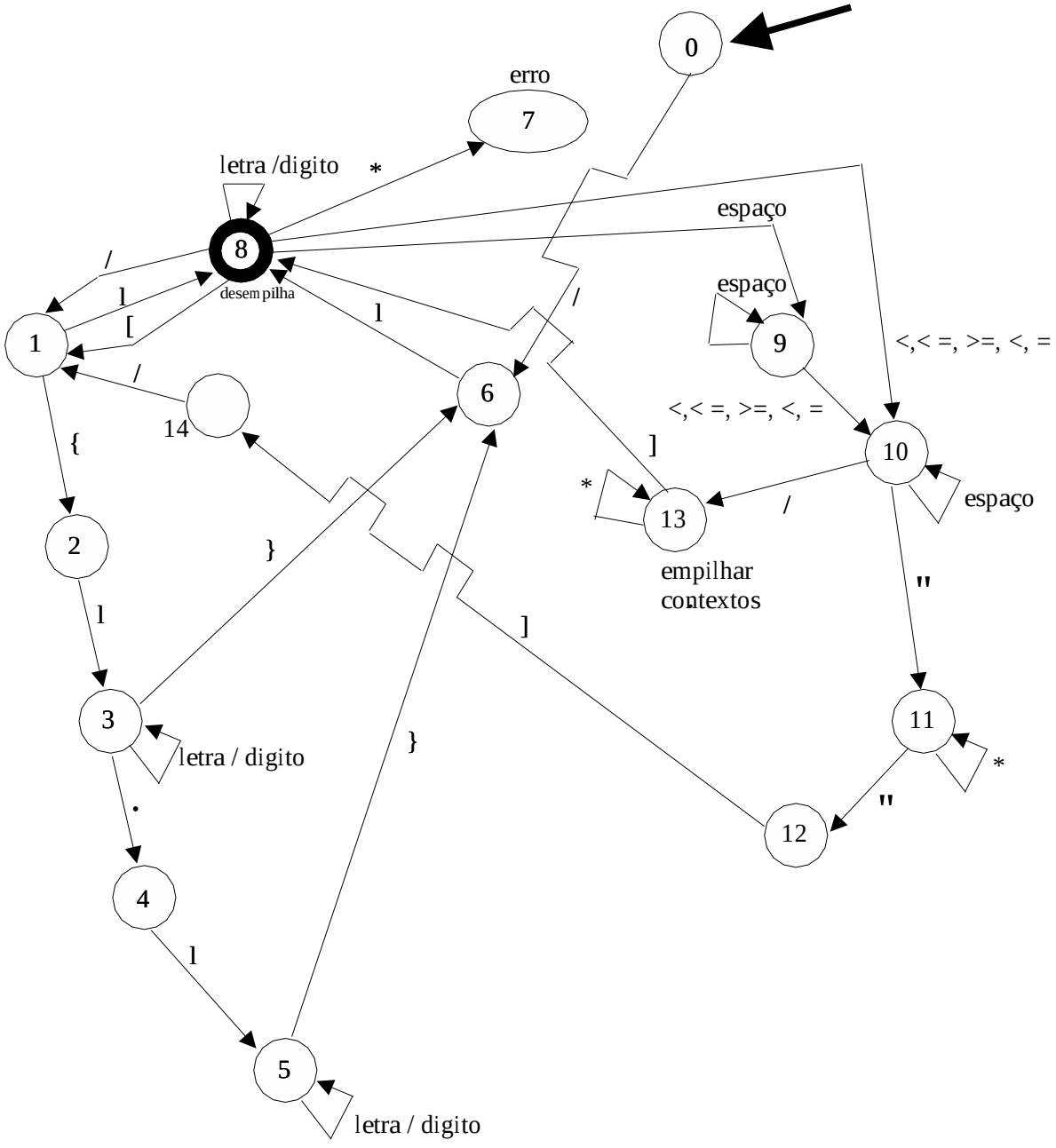


FIGURA C.1 – Autômato CXPath para Análise Léxica e tradução

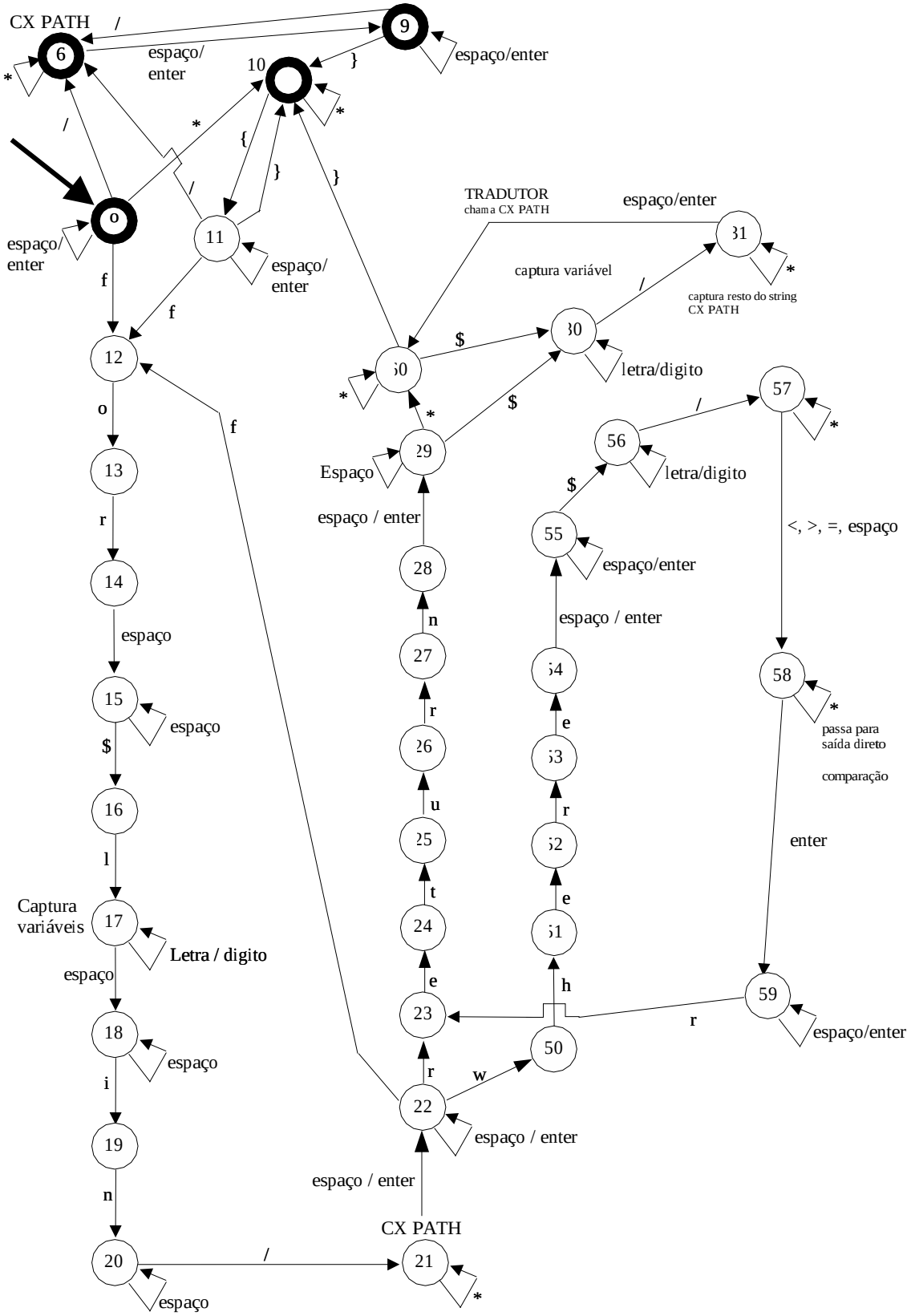


FIGURA C.2 – Autômato CXQuery para Análise Léxica e tradução

Bibliografia

[ABI 97] ABITEBOUL, S. et al. The Lorel Query Language for Semistructured Data. **International Journal on Digital Libraries** [S.l.], v.1, n.1, p.68–88, 1997.

[BAT 92] BATINI, C.; CERI, S.; NAVATHE, S. B. **Conceptual Database Design**: an entity-relationship approach. [S.l.]: Benjamin/Cummings, 1992.

[CAM 85] CAMPBELL, D. M.; EMBLEY, D. W.; CZEJDO, B. D. A Relationally Complete Query Language for an Entity-relationship Model. In: ENTITY-RELATIONSHIP APPROACH,4., 1985. **Proceedings...** [S.l.]: IEEE Computer Society, 1985. p.90–97. Disponível em: <http://dblp.uni-trier.de> Acesso em: 05 maio 2003.

[DAT 86] DATE, C. J. (Ed.). **Introdução a Sistemas de Bancos de Dados**. 3.ed. Rio de Janeiro: Campus, 1986. p.79 -82, p.319 - 324.

[DBL 2002] DBLP bibliography. Disponível em: *<http://www.informatik.uni-trier.de/ ley/db/>*. Acesso em: jul. 2002.

[DRA 2002] DRAFT, W. W. **Xquery 1.0 and XPath 2.0 Formal Semantics**. Disponível em: <http://www.w3.org/TR/query-semantics/> Acesso em: dez. 2002.

[ELM 99] ELMAGARMID, A.; RUSINKIEWICZ, M.; SHETH, A. M. **Management of Heterogeneous and Autonomous Database Systems**. [S.l.]: Morgan Kaufmann, 1999. 413p.

[FEG 2001] FEGARAS, L. **The OQL Algebra**. Texas: Department of Computer Science and Engineering, The University of Texas at Arlington. Disponível em: <http://www-cse.uta.edu/ fegaras/optimizer/oql-algebra.html.> Acesso em: 2001.

[FEG 2001a] FEGARAS, L. **The OQL Optimizer**. Texas: Department of Computer Science and Engineering, The University of Texas at Arlington. Disponível em: <http://www-cse.uta.edu/ fegaras/optimizer/oql-opt.html.> Acesso em 2001.

[HAL 98] HALPHIN, T. **Object-role modeling (orm/niam), handbook on architectures of information systems**. [S.l.]: Springer-Verlag, 1998. p.81–102.

[ISO 99] ISO. **ISO/IEC 9075** : Information Technology Database Language SQL. New York, 1999.

[LEN 2002] LENZ, E. **What's new in xpath 2.0.** Disponível em: <http://www.xml.com/pub/a/2002/03/20/xpath2.html.> Acesso em: mar. 2002.

[MAN 2001] MANOLESCU, I.; FLORESCU, D.; KOSSMANN, D. Answering xml queries over heterogeneous data sources. In: VERY LARGE DATABASES, VLDB, 2001, Roma, Italy. **Proceedings...** [S.l.:s.n.], 2001. p.241-250. Disponível em: < http://www-rocq.inria.fr/ manolesc/>. Acesso em: dez. 2002.

[MEL 2000] MELLO, R. S. **A Mediation Layer for Integration of XML Data Sources with Ontology Support**. 2002. PhD thesis proposal - Instituto de Informática, UFRGS, Porto Alegre.

[ODM 2002] ODMG, Object Data Management Coalition. **The OQL Query Language**. Disponível em:< http://www.odmg.org/.> Acesso em: 2001.

[OSZ 99] OSZU, M. T.; VALDURIEZ, P. **Distributed Databases:** Principles and Systems. [S.l.]: Prentice Hall, 1999.

[PAR 84] PARENT, C.; SPACCAPIETRA, S. An entity-relationship algebra. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 1. , 1984, Los Angeles, CA. **Proceedings. . .** [S.l.]: IEEE Computer Society, 1984. p.500–507.

[PRI 2001] PRICE, A. M. d. A.; TOSCANI, S. S. **Implementação de Linguagens de Programação**:compiladores. 2. ed. Porto Alegre: Sagra Luzzatto, 2001. 195p. (Livros didáticos,9).

[SAN 2001] SANTOS MELLO, R. dos. A Rule-based Conversion of a DTD to a Conceptual Schema. In: INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING, ER, 2001, Yokohama. **Conceptual Modeling** : proceedings. Berlin: Springer-Verlag, 2001. Disponível em: <http://www.inf.ufrgs.br/ ronaldo/publications.html.> Acesso em: 2002.

[SAN 2001a] SANTOS MELLO, R. dos. A Method for the Unification of XML Data. In: CONFERENCE ON OBJECT-ORIENTED PROGRAMMING SYSTEMS, LANGUAGES AND APPLICATIONS, OOPSLA, 16., 2001, Tampa Bay. [**Proceedings...**]. New York: ACM, 2001. Disponível em: <http://www.inf.ufrgs.br/ ronaldo/publications.html>. Acesso em: dez. 2002.

[SIG 99] SIGMOD record: xml version (v. 1.0). Disponível em: *<http://www.acm.org/sigmod/record/xml>*. Acesso em: dez. 1999.

[VID 2002] VIDAL, V. M. P.; BOAS, R. M. F. V. Uma Abordagem Top-Down para Geração das Correspondências entre XML Schemas Semânticos. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 17, 2002, Gramado. **Anais...** Porto Alegre: Instituto de Informática da UFRGS, 2002. p239-251.

[W3C 2001] W3C (Word Wide Web Consortium). Página Oficial do Consórcio. Disponível em: < http://www.w3c.org, W3c Home Page.>. Acesso em: dez. 2002.

[W3C 2002] W3C (Word Wide Web Consortium). **XML Query Languages**. Disponível em: <http://www.w3c.org/XML/Query> . Acesso em: dez. 2002.

[W3C 2002a] W3C (Word Wide Web Consortium). **Xquery 1.0**: An XML Query Language. Disponível em: <http://www.w3.org/TR/xquery.>. Acesso em: dez. 2002.

[W3C 2002b] W3C (Word Wide Web Consortium). **Introduction of XML Path Language** XPath 2.0. Disponível em:<*http://www.w3.org/TR/xpath20/#id-introduction*>. Acesso em: dez. 2002.

[W3C 2003] W3C. **Xquery Grammar**. Disponível em: <http://www.w3.org/2002/11/xquery-xpath-applets/xquery-bnf.html>. Acesso em: dez. 2002.

[XML 2001] XML Path Language - XPath 1.0. Disponível em: <*http://www.w3.org/TR/xpath*>. Acesso em: dez. 2001.

[XML 2002] XML Path Language (xpath) 2.0. **W3C Working Draft**. Disponível em: <http://www.w3.org/TR/xpath20/>. Acesso em: dez. 2002.