XML 응용시스템 개발을 위한 설계방안

김 경 수(金 敬 洙) · 주 경 수(朱 京 秀)
천안외국어대학 컴퓨터정보과 · 순천향대학교 공과대학 컴퓨터학부
전화 : (0417)550-0595 · (0418) 530-1318

A Design Methodology for XML Applications

Kim Kyung-Soo · Joo Kyung-Soo
Department of Computer Information, Chonan College of Foreign Studies
Department of Computer Science and Engineering, College of Engineering
Soonchunhyang University
E-mail : kkskim@mail.chonan-c.ac.kr · gsoojoo@sch.ac.kr

Abstract

Extensible Markup Language(XML) is fast emerging as the dominant standard for representing data in the World Wide Web. Sophisticated query engines that allow users to effectively tap the data stored in XML documents will be crucial to exploiting the full power of XML. While there has been a great deal of activity recently proposing new semi-structured data models and query languages for this purpose, this paper explores the more conservative approach of using traditional relational database engines for processing XML documents conforming to Document Type Descriptors(DTDs). In this paper, we describe how to generate relational schemas from XML DTDs. The main issues that must be addressed include (a) dealing with the complexity of DTD element specifications (b) resolving the conflict between the two-level nature of relational schemas (table and attribute) vs. the arbitrary nesting of XML DTD schemas and (c) dealing with set-valued attributes and recursion. We now propose a set of transformations that can be used to "simplify" any arbitrary DTD without undermining the effectiveness of queries over documents conforming to that DTD.

본 연구는 정보통신부의 대학 S/W 연구센터 지원사업에 의해 수행된 것임.

I. 서론

XML이 인터넷에서 데이터를 표현하는데 빠른 속도로 표준으로 잡아가고 있다[1]. HTML과 같이 XML도 SGML의 부분집합이며, HTML이 표현을 나타내는 언어인 반면에 XML은 데이터 그 자체를 나타낸다. XML에서 태그(tag)의 의미가 아주 중요하다. XML이 데이터 그 자체를 나타낸다는 것은 다음과 같은 3가지 의미가 있다. 첫째는 응용 프로그램이 다양한 방법으로 해석이 가능함을 의미하며, 둘째는 콘텐츠를 토대로 문서를 필터링할 수 있고, 셋째는 응용의 필요에 맞게 재구성할 수 있다.

XML 문서에 포함된 데이터는 XML DTD[2]를 수반함으로써 관계 데이터베이스 스키마를 도출할 수 있다. 그러나 이를 위해서는 몇 가지 해결해야 할 과제가 있는데 첫째는 복잡한 DTD 요소 사양을 해결해야 하며, 둘째는 관계형 스키마와 XML DTD 스키마 사

이의 충돌 문제 해결해야하고, 셋째는 집합값 속성과 순한 처리이다. 따라서 DTD를 단순화시킬 수 있는 DTD 변환 방안을 제시코자 했으며 XML DTD 요소를 관계 테이블로 대응시키는 방식에 따른 문제를 해결하기 위한 스키마 변환방식을 제안한다. 따라서 본 논문에서는 2절에서 스키마 변환방법을 소개하고 3절에서 결론과 추후 연구에 관하여 논한다.

II. 스키마 변환

본 절에서는 XML DTD로부터 관계형 데이터베이스 스키마를 도출키 위한 방안에 대하여 다루고 있다. 이를 위하여 해결해야 될 과제는

1) 복잡한 DTD 요소 사양 해결
2) 두 단계의 관계형 스키마(테이블과 속성)와 임의의 내포가 가능한 XML DTD 스키마 사이의 충돌 해소
3) 집합값 속성과 회귀 처리 등이다[3].

1. DTD 단순화

DTD에 부합된 문서들에 대한 질의의 효율 저하 없이, 임의의 DTD를 단순화시킬 수 있는 DTD 변환 방안들을 제시코자 한다. 본 변환방안은 [4]에서 제시한 유사한 변환방안들을 확장한 것이다.

1) 수평변환 : 내포성 정의를 플랫 표현으로 변경 (즉, 이진연산자들인 ","과 "|"은 임의의 연산자 안에 나타나지 않는다).
2) 단순화 변환 : 다수의 단항 연산자들은 하나의 단항 연산자로 단순화시킨다.
3) 그룹핑 변환 : 같은 이름을 갖는 부요소들을 구룹화한다.

본 변환방안에 따라 1) 일 대 다 , 그리고 2) null 또는 notnull 등의 의미는 보존되나, 요소들의 상대위치 정보 등은 상실된다. 다행히 상대위치 정보들은 특정 XML 문서가 관계스키마로 적재될 때 획득할 수 있다.

2. 스키마 변환방식

전통적으로 관계스키마들은 E-R (Entity Relationship) 모델과 같은 데이터 모델로부터 도출되어 왔다. 이는 개체들과 속성들이 명확하게 구분되기 때문에 각 개체와 그 속성들은 관계로 대응된다.

XML DTD를 관계로 전환시킬 때, DTD의 각 요소를 관계로 대응시키고 요소들의 속성들은 관계의 속성들로 대응시키는 경향이 있다. 그러나 DTD의 요소들과 속성들, 그리고 E-R 모델의 엔티티가 속성들 사이에 대응관계는 존재하지 않는다. 사실 XML에서 firstanme과 lastname이 attribute이라면, name 아래 내포될 수 없다. 왜냐하면, XML attribute는 내포된 구조를 갖지 못하기 때문이다. 이와 같이 요소들을 관계로 직접 대응시키는 방식은 XML 문서에 대한 과도한 분할을 야기할 수 있다.

제안하는 변환방식은, 가능한 한 한 요소의 자손들을 하나의 관계로 묶음으로써 과다한 분할 문제를 해결코자 한다. 그러나 본 방식은 XML 문서는 DTD에서의 임의의 요소에 대하여 루트될 수 있기 때문에, 이에 따라 모든 XML 요소들에 대하여 관계들을 생성한다. 예를 들면, 그림 1의 author 요소는 firstname, lastname, 그리고 address를 속성으로 하는 관계로 대응시킬 수 있다. 이제 두 개의 과제인 집합값 속성과 회귀에 대하여 다루어 보자. 그림 2의 DTD에서 article을 위한 관계를 만들 때 기존의 관계 모델이 집합값 속성을 지원하지 않기 때문에, author들의 집합을 처리할 수 없다. 그 대신 RDBMS에서 집합을 처리하기 위한 표준 방식에 따라 author를 위한 관계를 만들고 외래키를 이용하여 author에서 article으로의 연결을 도모한다. 단순히 인라인만을 사용하는 것은 회귀에서의 내포를 제한시킨다. 따라서 관계키의 개념을 이용하여 회귀관계성을 표현하고, 아울러 회귀관계성을 도출하기 위하여 관계적 회귀처리를 수행한다. 이를 위하여 DTD 그래프 개념을 도입한다.

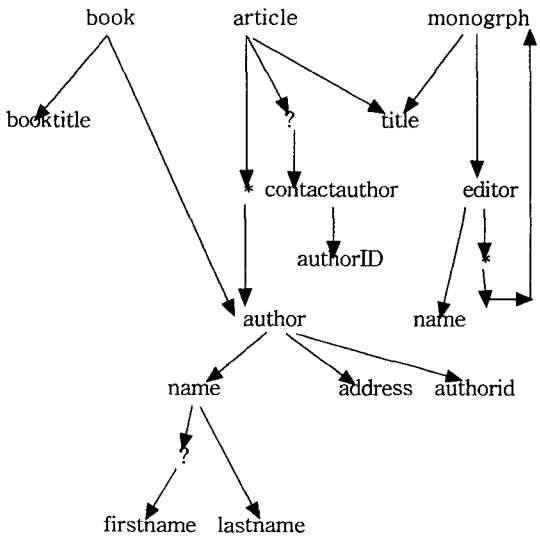
DTD 그래프는 DTD의 구조를 나타내어, 각 노드들은 DTD 내의 attribute들과 연산자들이다. 각 요소는 그래프에서 정확히 한번만 나타나는 반면에, attribute들과 연산자들은 DTD에 있는 만큼 나타나게 된다. 그림 2의 DTD에 대응되는 DTD 그래프가 그림 3이다. 한편 DTD 그래프에서의 사이클은 DTD에서 회귀관계성이 있음을 보여준다.

```
<book>
<booktitile> 노자의 21세기 </booktitle>
<author id "도올>
        <name>
                <firstname>김</firstname>
                <lastname>용옥</lastname>
        </name>
        <address>
                <city>서울시</city>
                <zip>151</zip>
        </address>
</author>
</book>
```

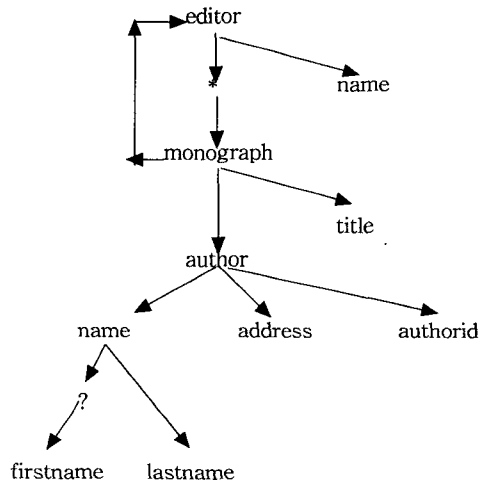
[그림 1] XML문서

```
<!ELEMENT book (booktitle,author)>
<!ELEMENT article (title,author*,contactauthor>
<!ELEMENT contactauthor EMPTY>
<ATTLIST contactauthor author IDREF IMPIED>
<!ELEMENT monograph(title,author,editor)>
<!ELEMENT editor (monograph*)>
<!ATTLIST editor name CDATA #REQUIRED>
<!ELEMENT author(name,address)>
<!ATTLIST author id ID #REQUIRED>
<!ELEMENT name (firsname?,lastname)>
<!ELEMENT firstname(#PCDATA)>
<!ELEMENT lastname(#PCDATA)>
<!ELEMENT address ANY>
```

[그림 2] book DTD



[그림 3] DTD 그래프



[그림 4] 요소 그래프

DTD에 따라 생성된 스키마는 각 요소에 대하여 생성된 관계집합의 합집합이다. 특정 요소에 대하여 관계집합을 생성하기 위하여 요소그래프(element graph)라
부르는 그래프 구조를 도출한다. 요소그래프는 다음과 같이 만든다.

우선 DTD 그래프에 대하여, 관계를 만들고자 하는 요소 노드에서 시작하는 depth first 탐색을 한다. 각 노드는 처음 방문되었으면, "visited"로 표시하고, 아울러 전 자식 노드들이 전부 방문되었으면 unmark한다. 만약 DTD 그래프에서 unmark된 노드가 depth first 탐색 중 방문되었다면, 같은 이름을 갖는 새로운 노드가 요소 그래프에 생성된다. 더욱이 현재의 DTD 노드의 DFS 부모와 같은 이름을 갖는, 가장 최근에 생성된 요소 그래프의 노드에서 방금 생성된 노드로 규칙적 에지를 생성시킨다. 이미 표시된 DTD 노드에 대하여 탐색코자 시도가 발생했다면, 요소 그래프에서 가장 최근에 생성된 노드로부터 표시된 DTD 노드와 같은 이름을 갖는 요소 그래프의 가장 최근에 생성된 노드로 backpointer 에지를 생성시킨다. 그림 3의 DTD 그래프에서 editor 요소를 위한 요소 그래프가 그림 4에 있다. 직관적으로 요소 그래프는 DTD 그래프의 관련된 부분을 트리 구조로 확장한다. 일단 요소 그래프가 주어지면, 관계들은 다음과 같이 만들어진다. 하나의 관계가 요소 그래프의 루트 요소에 대하여 생성된다. 루트 요소의 모든 자손들은 해당 관계에 해석되어지나, (a) * 노드의 직계자식들은 별도의 관계들을 만들어 처리한다-이는 집합값을 갖는 자식을 위해 별도의 관계를 만들기 위함이다. (b) 자신에 대한 backpointer 에지를 갖는 각 노드들은 별도의 관계로 처리한다-이는 회귀관계성을 처리하기 위하여 별도의 관계를 생성하는 것이다.

그림 5은 그림 2의 DTD로부터 생성된 관계 스키마를 보여준다. 본 스키마에서 몇가지 주목할만한 점들이 보인다. 관계들에서 속성들은 그 관계의 루트요소로부터의 통로에 의해 명명되어져 있고, 각 관계는 키 역할을 하는 ID 필드를 갖는다. 부모를 갖는 요소 노드들에 대응하는 모든 관계들은 외래키 역할을 하는 parentID 필드를 갖고 있다. 예를 들면, article. author 관계는 article들을 author들과 죠인시키는 외래키 article.author.parentID를 갖는다.

변환방식이 "책의 저자를 보여라"와 같은 형태의 질의들에 대해서는 좋지만, 다른 형태의 질의에 대해서는 효율적이지 못하다. 예를 들면, "성이 김인 저자 모두를 보여라"와 같은 질의는 5개의 별개의 질의들의 합집합으로 실행되어야 한다. 제시된 변환방식의 또

book(bookID:integer,book.booktitle:string,book.author.name.firstname:string,book.author.name.lastname:
string,book.author.address:srting, aurhor.authorid:string)

booktitle (booktitleID : integer,booktitle:string)

article(articleID : inreger, article.contactautor.ahthorid:string,article.title:string)

article.author(article.authorID : inreger, article.author.parentID: integer,article.author.name.firstname:
string, article.author.name.lastname : string,
article.ahthor.adress : string, article.author.authorid:string)

contactautor (contactautorID : ,integer,contactauhor. authorid : string)

title (titleID : integer, title:string)

monograph(monographID: integer.monograph.parentID:integer.monograph.title:string,
editor.monograph.author.name.firstname:string,editor.monograph.author.name.lastname:string,
editor.monograph.author.adress.:string,editor.monograph.author.authorid:string)

editor.monograph(editor.monographID:integer.editor.monograph.parentID:integer.editor.monograph.title:
string,editor.monograph.author.name.firstname:string,editor.monograph.author.name.lastname:string,
editor.monograph.author.adress.:string,editor.monograph.author.authorid:string)

author(authorID:integer,author.name.firstname:string,author.name.lastname:stnng,author.address: strong, author.authorid:string)

name (nameID: Integer, name.firstname: string, name.lastname: string)

firstname (firstnameID: integer, :firstname: string)

lastname (lastnameID: inteser, lastname: string)

address (addressID: Integer, address: string)

[그림 5] DTD로부터 생성된 관계 스키마

다른 단점은 너무 많은 관계들을 생성시킨다는 점이다.

Ⅲ. 결론 및 추후 연구

HTML이 단순히 표현을 나타낸다면 XML은 데이터 그 자체를 나타낸다. 이것은 응용 프로그램이 다양한 방법으로 해석이 가능하다는 것, 또한 태그내에 있는 정보를 필터링할 수 있고, 데이터를 재구성할 수 있다는 것이다. 데이터베이스 관점에서 XML은 XML 문서에 포함된 데이터를 DTD 형식에 따라 관계 데이터베이스 스키마 변환이 가능하다는 것이다. 그러나 몇가지 문제가 있는데, 복잡한 DTD 요소 내용을 단순화시킬 필요가 있으며, 관계형 스키마와 XML DTD 사이의 충돌 문제를, 관계 데이터베이스에서 제공되지 못하는 집합값 속성과 회귀 처리 문제를 해결해야 한다. 따라서 본 논문은 DTD를 단순화시킬 수 있는 DTD 변환 방안을 제시코자 했으며, XML DTD 요소를 관계 테이블로 대응시키는 방식에 따른 문제 해결을 위한 스키마 변환 방안을 제안했다. 추후, 변환방안의 문제점을 보안하여 좀 더 효율적인 변환 방안에 대한 연구가 필요하다.

Ⅳ. 참고문헌

[1] T. Bray, J. Paoli, C. M. Sperberg-McQueen, "Extensible Markup Language(XML) 1.0", http://www.w3.org/TR/REC-xml

[2] J. Bosak, T. Bary, D.Connolly, E.Maler, G. Nicol, C. M. Sperberg-McQueen,L. Wood, J.Clark, "W3C XML Specification DTD", http://www.w3.org/XML/1998/06/xmlspec-report-19980910.htm

[3] Jayavel Shanmugasundaram, Kristin Tufte "Relational Databases for Querying XML Documents: Limitations and Opportunities." Proceedings of the 25th VLDB Conference Edinburgh, Scotland, 1999

[4] Deutsch, M. Fernandez, D .Suciu, "Storing Semi-structured Data with STORED", Procedings of the ACM SIGMOD Conference, Philadelphia, Pennslyvania, May 1999