XML 문서의 공통구조를 이용한 효율적인 릴레이션 스키마 추출기법

안성은[0], 이정선, 최황규
강원대학교 전기전자정보통신공학부
{bomber[0], sunny}@mail.kangwon.ac.kr, hkchoi@kangwon.ac.kr

An Efficient Relational Schema Extracting Technique Using Common Structure in XML Documents

Sung-Eun Ahn[0] Jung-Sun Lee Hwang-Kyu Choi
Dept. of Computer and Electrical Engineering, Kangwon National University

요 약

XML은 웹 상에서 데이터를 표현하고 교환하기 위한 표준으로 등장하고 있다. 최근에 웹 상에서 다루어지는 데이터의 양이 급격하게 증가함에 따라 데이터의 형태는 구조적인 릴레이션 데이터에서 반 구조적인 데이터에 이르기까지 다양하다. 앞으로 웹에서 반 구조적 데이터를 대표할 XML 문서들이 많아지면 그 데이터들간의 의미적 구조적 관계를 설정하는 스키마를 추출하여 그에 따라 데이터를 구조화 시켜 정보로써의 가치를 만들 수 있는 새로운 저장 기법들이 필요하다. 본 논문에서는 XML 문서의 DTD를 이용하여 동일한 DTD를 사용하는 XML 문서들의 공통구조를 추출하여 관계 데이터베이스 시스템에 XML 문서를 저장하기 위한 릴레이션 스키마 추출 기법을 제안한다.

1. 서 론

XML[1]은 W3C에 의해 제안된 SGML의 한 부분집합으로써 HTML과 SGML의 단점을 보안한 웹 상에서 데이터를 교환하기 위한 표준 언어이다. XML은 DTD(Document Type Definition)를 통해서 문서자체에 문서의 구조를 기술하고 있다[1]. 문서의 구조를 사용자가 원하는 대로 정의할 수 있으며, 이러한 구조적 유동성은 다양한 형태의 데이터를 XML로 표현될 수 있게 해준다. 즉, 웹에서 운용되는 데이터가 동일한 형태로 저장, 처리 될 수 있음을 의미한다. 앞으로 웹에서 반 구조적 데이터를 대표할 XML 문서들이 많아지면 그 데이터들간의 의미적, 구조적 관계를 설정하는 스키마를 추출하여 그에 따라 데이터를 구조화 시켜 정보로써의 가치를 만들 수 있는 새로운 저장 기법들이 필요하다.

본 논문에서는 XML 문서의 DTD를 이용하여 동일한 DTD를 사용하는 XML 문서들의 공통 구조를 추출한 후 릴레이션 데이터 베이스 시스템에 XML 문서들을 저장하기 위한 스키마를 추출하는 기법을 제안한다. 본 논문은 2장에서 관련연구를 살펴보고, 3장에서는 스키마 추출기법을 제안한다. 마지막으로 4장에서 결론을 맺는다.

2. 관련연구

기존의 릴레이션 데이터베이스에 XML 데이터를 저장하기 위한 스키마를 추출방법에 관한 여러 연구들이 있었다. 먼저, [2]는 XML데이터의 element와 attribute를 구분하지 않고 element와 value를 하나의 테이블에 저장하는 방법과 element와 value값의 특성에 따라 여러 개의 테이블로 나누어 저장하는 방법을 제안했다. 이는 릴레이션 테이블에 XML 데이터를 저장하는 가장 기본적인 방법으로 볼 수 있다.

[3]은 XML 데이터를 Element, Attribute, Text, Path 네 가지 릴레이션으로 나누어 저장하는 방법을 제안했다. 이 방법은 릴레이션 스키마가 XML문서의 DTD와 독립적이므로, XML문서의 스키마 역할을 하는 DTD를 이용할 수 없으며, DTD의 중요한 요소인 IDREFS를 고려하지 않고 있다. [4]는 XML 데이터를 처리하는데 있어 릴레이션 데이터 처리 방법과 반 구조적 데이터 처리 방법을 혼합했다.

[5]는 기존의 문서 중심의 접근 방법과는 달리 XML문서의 DTD를 기반으로 릴레이션 스키마를 추출하는 기법을 제안했다. DTD 구조를 이용함으로써 가능한 적은 수의 테이블로 XML 데이터를 맵핑시키기 위한 Hybrid Inlining Algorithm을 제안했다. [5]의 방법은 DTD의 노드구조에 다수의 '*' '?' 연산자가 발생할 경우 릴레이션 테이블에 다수의 널(NULL) 값이 존재한다는 단점이 있다. 본 논문에서는 이러한 단점을 극복하면서 릴레이션 테이블 수를 줄일 수 있는 기법을 제안한다.

3. DTD를 이용한 절대 스키마 추출 기법

XML 문서는 XML 데이터들의 스키마 역할을 하는 DTD를 가지고 있다. 즉, DTD는 XML 문서의 구조와 의미를 정의한 것으로써, DTD가 강제적인 요소는 아니지만 도서목록이나 은행의 고객관리정보와 같은 데이터를 XML 문서로 만들 때에는 문서의 구조에 관한 정보를 가지고 있는 DTD가 반드시 필요할 것이다. 앞으로 웹 상에서 다루어지는 데이터가 XML 형태로 표현된다면 특정분야의 데이터를 위한 일반화된 DTD가 필요할 것이며 각각의 수많은 XML 문서들은 이러한 일반화된 DTD를 기반으로 작성 될 것이다.

본 논문은 XML문서의 구조 정보를 담고 있는 DTD(Document Type Definition)를 이용하여 동일한 DTD를 사용하는 XML 문서들의 공통 스키마를 추출하는 기법을 사용하여 릴레이션 스키마를 추출한다.

3.1 기본 아이디어

XML 문서의 DTD(Document Type Definition)의 요소(element) 선언부에는 각 노드와 자식노드의 발생 빈도를 결정해주는 카디날리티 연산자(Cardinality Operator)를 가지고 있다[1]. 본 논문에서는 Cardinality Operator를 CO 라고 약칭하며, 이 CO를 이용해 각 노드간의 발생 빈도 범위를 설정하여 DTD 그래프에 적용함으로써 XML 문서들의 공통 스키마를 추출한다.

3.2 카디날리티 연산자(Cardinality Operator)

표1에서 (none)은 노드가 단 한 개만 존재하고, '?'는 있거나 없거나 둘 중 한가지 경우이다. '*'는 없거나 여러 개 존재하고, '+'는 한 개 이상 여러 개 존재한다는 의미이다. CO를 부모노드에서 자식노드까지 적용시키기 위해서 표2와 같은 아이디어를 냈다. 즉, 표2와 같이 부모와 자식노드의 '합 연산자'를 표현 할 수 있다. 각 연산자의 합 연산을 표시한 것으로 괄호 안의 숫자들은 노드의 발생 범위를 나타낸다. 예를 들어 다음과 같은 DTD가 있다.

```
<!ELEMENT 연구실목록(교수, 학생+)>
<!ELEMENT 학생(주소?, 전화*)>
```

여기서 "학생/주소"에 대한 합 연산자는 +? → * 이다. 즉, "학생/주소"은 (0,1,2,3,...)의 노드 개수 범위를 가진다. 다시 말하면 "학생"이

라는 노드는 한번이상 여러 번 문서에 발생할 수 있지만 "학생/주소" 라는 노드는 문서에 발생하지 않을 수 도 있다는 가능성을 가진다. 위와 같은 방법으로 루트노드에서 어떤 위치에 있는 자식노드까지의 노드 개수 범위를 'n, ?, *, +'로 나타낼 수 있다. 이러한 방법을 전체 DTD그래프의 루트 노드부터 마지막 리프 노드까지 적용하면 'n'과 '+' 연산자를 가지는 노드는 동일한 DTD를 바탕으로 작성된 어떠한 XML 문서에도 반드시 존재하는 노드가 된다.

표 1. 카디날리티 연산자

기호	노드의 개수 범위	예
(none)	(1)	ELEMENT A
?	(0,1)	ELEMENT A?
*	(0,1,2,3,...)	ELEMENT A*
+	(,1,2,3,4,...)	ELEMENT A+

표 2. 카디날리티 연산자 변환

(none)을 n으로 표기	
nn → n (1)	** → * (0,1,2,3,...)
n? → ? (0,1)	*n → * (0,1,2,3,...)
n* → * (0,1,2,3,...)	*? → * (0,1,2,3,...)
n+ → + (1,2,3,4,...)	*+ → * (0,1,2,3,...)
?? → ? (0,1)	++ → + (0,1,2,3,...)
?n → ? (0,1)	+n → + (1,2,3,4,...)
?* → * (0,1,2,3,...)	+? → * (0,1,2,3,...)
?+ → * (0,1,2,3,...)	+* → * (0,1,2,3,...)

3.3 CO의 속성 값 적용

표 3에서와 같이 XML 문서의 DTD 속성 선언부에는 각각의 속성 값들의 필수 존재 여부를 결정해주는 속성 기본 파라미터 값들이 있다. #REQUIRED는 속성 값이 반드시 존재해야 한다는 의미이므로 CO의 'n'으로 표현 될 수 있으며, #IMPLIED와 #FIXED는 존재할 수도 있고 없을 수도 있다는 의미이므로 CO의 '?' 로 표현 될 수 있다.

DTD의 속성의 타입 부분에서 주의 깊게 보아야 할 부분은 ID와 IDREF IDREFS이다. ID는 각각의 엘리먼트 노드들의 식별자 역할을 하는 요소로써 IDREF와 IDREFS를 속성으로 가지는 엘리먼트 노드들의 레퍼런스 역할을 한다. IDREF 요소는 속성 값으로 ID값을 하나 가질 수 있고, IDREFS 요소는 속성 값으로 여러 개의 ID값을 공백으로 구분하여 가질 수 있다. 그럼으로 IDREFS는 CO의 +으로 표현 될 수 있다. 즉, IDREFS는 속성 값의 노드이지만, 엘리먼트 노드가 '+' 연산자를 가지고 있는 것 과 같은 효과를 나타내기 때문에 IDREFS 요소를 스키마 형태로 나타내기 위해서는 엘리먼트 노드처럼 처리해야 한다.

표 3. 속성 파라미터 값

속성기본파라미터	설명	표기
#REQUIRED	속성은 반드시 요소의 모든 인스턴스 안에 존재	n
#IMPLIED	옵션, 있거나 없을 수있다	?
#FIXED	있을 수도, 없을 수도 있지만 있다면 기본 값을 가짐	?

3.3 절대 스키마

그림 1은 본 논문에서 사용할 '연구실목록.dtd' 이다. 이 DTD를 directed labeled graph를 그린 후에 연산자를 적용하면 그림 2와 같다. 그림 2에 합 연산자를 적용하여 그래프를 그리면 그림 3과 같으며 각각의 CO는 XML 문서의 노드 범위를 나타낸다. 즉, '+'로 표시된 노드는 문서 내에서 1번 이상 나타나고, '*'로 표시된 노드는 0번 이상 나타날 것이다. 즉, 그림 3에서와 같이 '+'와 'n'연산자를 가지는 노드들을 따로 분리해서 그래프를 그리면 그림 4와 같다. 이러한 그래프를 "절대 스키마 그래프"라 칭한다. 절대 스키마는 전체 XML 문서들에서 반드시 존재하는 노드들만으로 구성된 것으로, 절대 스키마를 바탕으로 관계 데이터베이스에 저장하기 위한 릴레리션 스키마(relation schema)를 추출한다.

```
<!ELEMENT 연구실목록 (교수, 학생+, 연구실, 프로젝트+)>
<!ELEMENT 교수 (이름, 소속, 메일, 주소?, 전화*, 경력사항)>
<!ELEMENT 경력사항 (학력, 연구논문*, 이력*)>
<!ELEMENT 학력 (학사, 석사, 박사)>
<!ELEMENT 학사 (#PCDATA)>
<!ELEMENT 석사 (#PCDATA)>
<!ELEMENT 박사 (#PCDATA)>
<!ELEMENT 연구논문 (#PCDATA)>
<!ELEMENT 이력 (#PCDATA)>
<!ELEMENT 학생 (이름, 소속, 메일, 학년, 주소?, 전화*)>
<!ELEMENT 학년 (#PCDATA)>
<!ATTLIST 학생
        학번 ID #REQUIRED>
<!ELEMENT 연구실 (소속, 소개?, 주소, 전화*)>
<!ELEMENT 소개 (#PCDATA)>
<!ELEMENT 프로젝트 (제목, 요약문?, 시작일?, 종료일?)>
<!ELEMENT 제목 (#PCDATA)>
<!ELEMENT 요약문 (#PCDATA)>
<!ELEMENT 시작일 (#PCDATA)>
<!ELEMENT 종료일 (#PCDATA)>
<!ATTLIST 프로젝트
        연구원 IDREFS #REQUIRED>
<!ELEMENT 이름 (#PCDATA)>
<!ELEMENT 소속 (#PCDATA)>
<!ELEMENT 메일 (#PCDATA)>
<!ELEMENT 주소 (#PCDATA)>
<!ELEMENT 전화 (#PCDATA)>
```

그림 1. 연구실목록 DTD

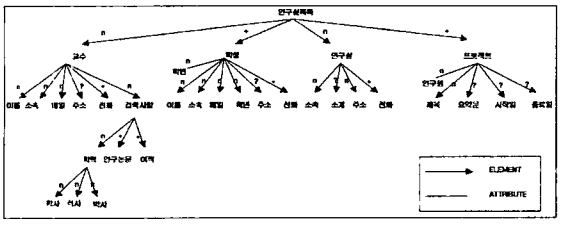

그림 2. DTD 연산자(CO) 그래프

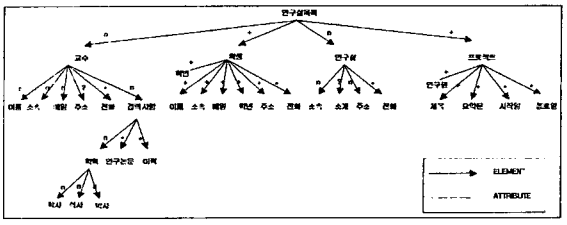

그림 3. 합 연산자(CO) 그래프

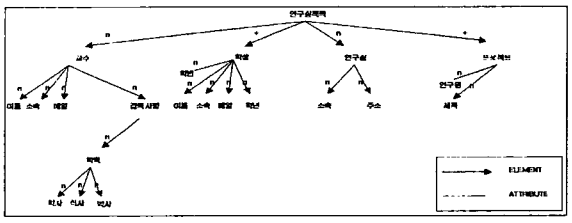

그림 4 절대 스키마 그래프

3.4 비 절대 스키마

그림 5는 DTD 그래프에서 절대 스키마를 제외한 나머지 노드들을 나타내는 것으로 데이터의 손실을 막기 위해서 본고에서는 나머지 스키마를 절대 스키마와 분리하여 "비 절대 스키마"라 칭하며, 관계 데이터베이스 시스템에 저장하기 위한 스키마를 따로 추출한다.

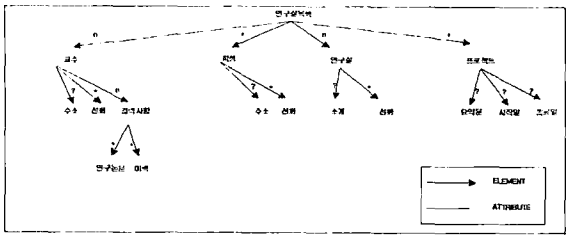

그림 5. 비 절대 스키마 그래프

3.5 절대 릴레이션 스키마 추출

초기 DTD의 CO구조는 [5]에서 사용된 기법을 통하여 간단하게 된 상태이며, ENTITY나 NOTATION 같은 DTD 구성 요소는 고려하지 않았다. 이때 절대 릴레이션 스키마 추출 과정은 다음과 같다.

1) DTD 그래프 생성 : DTD의 유효성을 검증하여 초기화된 DTD를 통해 각각의 노드정보를 가져온다.
2) 합 연산자 그래프 생성 : DTD내의 CO 정보를 추출하여 합 연산자를 생성한다.
3) 절대 스키마 생성 : CO 정보를 통해 절대스키마를 생성할 노드를 불리 한다.
4) 비 절대 스키마 생성 : 절대 스키마 외의 나머지 노드들을 따로 생성한다.
5) 테이블 노드 생성

절대 스키마에서 '+'연산자를 가지는 노드는 하나의 테이블로 맵핑이 되며, 부모와 자식 테이블 노드와는 foreign key로 참조된다. 테이블 노드의 자식노드들은 릴레이션 테이블의 속성 값이 된다. 또한, 속성 기본 파라미터 값이 'IDREFS'인 노드 또한 하나의 테이블로 맵핑되며 부모 노드와 foreign key로 참조된다. 각각의 테이블노드에 대한 primary key 필드를 생성한다. ID 속성이 존재하면 사용하며, 없는 경우는 인적으로 생성한다.. '@'표기는 속성 값을 표현하기 위함이다. 공유되는 엘리먼트를 위해 릴레이션 테이블에 parent field와 root field 생성한다. 그림 6은, "데이터베이스.xml" "멀티미디어.xml" "인공지능.xml", 3개의 XML문서를 기반으로 생성된 절대 릴레이션 스키마이다. 절대 릴레이션 스키마는 NULL값을 포함하지 않으며 전체 XML 문서들에 공통적으로 존재하는 노드들만으로 구성 된 것이다.

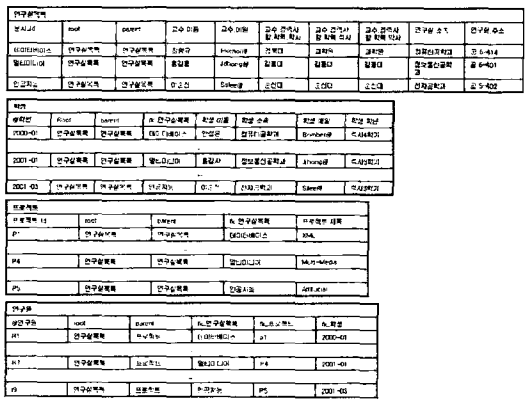

그림 6. 절대 릴레이션 테이블

3.6 비 절대 릴레이션 스키마 추출

비 절대 스키마에 포함되는 데이터들은 절대 스키마 노드들의 자식 노드들로 구성된다. 즉, 비 절대 릴레이션 스키마는 그림 7과 같이 각각의 절대 릴레이션 테이블을 foreign key로 참조하고, root, parent, element name, attribute name, value field를 생성한다.

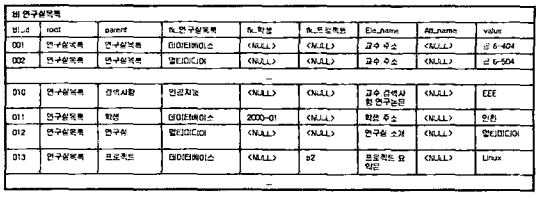

그림 7. 비 절대 릴레이션 테이블

4. 결 론

표 4는 본 논문에서 사용한 DTD의 구성을 보여주며, 표 5는 기존의 Hybrid Inlining 방법과 본 논문에서 제안한 절대스키마 추출 기법과의 테이블 수를 나타낸다. Inlining방법과 달리 XML문서의 공통구조를 따로 추출하기 때문에 DTD구조에서 '*'와 '?' 연산자가 나타내는 NULL 값의 단점을 극복하였다.

XML 문서를 릴레이션으로 맵핑하는 방법에 있어서는 DTD의 구조가 가장 중요한 역할을 한다. 기존의 방법은 DTD에 '*' '?' 연산자가 많이 존재한다면, 각각의 테이블은 많은 수의 NULL값을 가지게 되며 이는 성능저하와 비용 증대로 이어진다. 본 논문에서 제안한 절대스키마는 이러한 점을 극복하기 위한 것이다. 하지만 DTD 설계자가 상위 노드에 중요한 데이터를 '+'와 'n' 연산자가 아닌 '*'와 '?' 연산자로 설계한다면, DTD 그래프의 CO 적용에 있어 Level을 고려하여야 하며 이러한 과정은 또한 앞에서 언급한 단점들을 포함할 수 있다. 앞으로 시스템 구성을 통한 기존의 여러 방법들과의 비교 분석이 필요할 것이다.

표 4. 연구실목록 DTD 구성

element	attribute
24	2
'+' : 2개 '*' : 3개	IDREFS : 1개

표 5. 릴레이션 테이블 비교

	Hybrid Inining	절대스키마추출기법
테이블 수	7	5
	'+노드' + '*노드' + 'IDREFS노드' + 1(root)	'+노드' +'IDREFS노드' + 2(root + 비 절대스키마)
테이블 속성 수	53	38
널 값 존재 테이블 수	4	1

참고문헌

[1]T. Bray, J. Paoli, and C. M. Sperberg-McQueen, "Extensible Markup Language(XML)1.0," http://www.w3.org/TR/REC-xml, W3C Recommendation 6, October 2000.

[2] D. Florescu and D. Kossmann, "Storing and Querying XML Data Using an RDBMS," IEEE Data Eng. Bulletin, 1999.

[3] T. Shimura, M. Yoshikawa, and S. Uemura, "Storage and Retrieval of XML Documents Using Object-Relational Databases," DEXA 1999.

[4] A. Deutsch, M. F. Fernandez, D. Suciu, "Storing Semistructured Data with STORED," ACM SIGMOD Conference 1999.

[5] J. Shanmugasundaram, H. Gang, K. Tufte, C. Zhang, D. J. DeWitt, and J. F. Naughton, "Relational databases for querying XML documents: Limitation sand opportunities," In VLDB '99, Edinburgh, Scotland, 1999.