XML DTD로부터 관계형 테이블로의 사상구조 설계

박은경* 정채영* 김현주** 배종민*
*경상대학교 컴퓨터과학과 **경남정보대학 컴퓨터정보시스템계열
{pek, wcdi}@rtp.gsnu.ac.kr khj@kit.ac.kr jmbae@nongae.gsnu.ac.kr

Design of a Mapping Structure of XML DTDs to Relational Tables

Eun-Kyung Park* Chai-Young Jeong* Hyun-Ju Kim** Jong-Min Bae*
*Dept. of Computer Science, Gyeongsang National University
**Group of Computer Information, Kyung Name College of Information & Technology

요 약

데이터로서 XML 역할에 대한 중요성이 증대되어감에 따라, 구조 정보를 내포하고 있는 데이터로서의 XML 문서를 효과적으로 관리하는 구조설계 및 질의어 처리에 대한 연구가 활발히 진행되고 있다. XML 저장구조는 XML 스키마에 의존하지 않고, 반구조적 데이터에 대한 저장과 질의처리 설계방법과 DTD와 같은 XML 스키마를 기반으로 저장구조를 설계하는 방법이 있다. DTD는 XML 문서의 논리적인 구조정보를 표현하는 역할을 가지고 있으며, 그러한 역할은 이기종간의 문서교환시 더욱 중요해진다. DTD 기반의 XML문서를 관계형 테이블로 사상하고자 할 때, 사상과정이 자동적으로 이루어지는 경우와 사용자나 관리자가 직접 테이블 생성에 관여하는 수동적인 사상방법이 있다. 수동적인 사상과정은 질의처리기 구현시 데이터베이스 설계 방법에 크게 의존하게 되는 단점이 있지만, 사용자가 주어진 DTD 구조에 따라서 특정 응용에 알맞는 저장구조를 직접 설계하기 때문에 더 효율적인 저장구조를 설계할 수 있다. 본 논문에서는 관계형 DB를 질의처리 하기 위한 하부 저장소로 사용하고, DTD 기반의 XML 문서를 관계형 DB 테이블간의 수동적인 사상과정을 통하여 효율적인 XML 문서 저장 구조를 설계하는 방법을 제시한다.

1. 서론

XML은 문서에 대한 구조정보를 제공할 뿐만 아니라, XML 태그는 데이터를 해석하는데 사용할 수 있기 때문에, 데이터로서 XML 역할에 대한 중요성이 인식되어 왔다. 이에 따라서 구조 정보를 내포하고 있는 데이터로서의 XML 문서를 효과적으로 관리하는 구조와 질의어 설계 및 처리에 대한 연구가 많이 진행되고 있다.

XML 문서를 관리하고 질의하는 방법으로는, Lorel[3]이나 STORED[1]와 같이 기존의 XML 스키마에 의존하지 않고 반구조적 데이터에 대한 저장과 질의처리 메커니즘을 개발하는 방향이 있고, XML 스키마를 기반으로 저장구조를 설계하는 방향이 있다. XML 스키마로는 DCD, XML Schema, DTD 등이 있다. 이 중에서 DTD는 이기종간의 문서교환에서 중요한 역할을 하기 때문에 DTD기반의 XML 문서 관리 시스템에 관한 연구는 현실적으로 중요한 위치를 차지한다.

XML 문서를 저장하고, 질의처리를 하기 위한 하부 저장소로서 파일 시스템, 기존의 RDBMS, OODBMS, 그리고 반구조적 데이터에 대한 고유의 시스템을 바탕으로 설계될 수 있다.[2] 이 중에서 관계형 데이터베이스에서 DTD기반의 XML 문서 저장 구조를 설계하기 위해서는 DTD를 데이터베이스 테이블로 사상시켜야 한다. 이러한 사상과정은 자동으로 이루어질 수도 있고, 수동으로 이루어질 수도 있다. 테이블 생성이 수동으로 이루어지는 경우에는 사용자가 테이블 생성에 관여를 해야 하기 때문에 질의처리의 효율성이 사용자의 데이터베이스 설계 방법에 크게 의존하게 되는 단점이 있다. 그러나, 문서의 저장 구조를 설계할 때, 복잡한 DTD를 단순한 형태로 변형시킬 필요가 없기 때문에 데이터베이스에 저장된 XML 문서로부터 원래의 XML 문서로 그대로 복원하기가 쉽고, 문서의 구조를 파악하기 위하여 시스템내에 DTD 그래프를 생성시킬 필요가 없으며, 재귀적 종속관계에 있는 엘리먼트를 찾는 과정에 대한 처리가 필요없다. 따라서, 사용자는 주어진 DTD 구조에 따라서 특정 응용에 알맞는 저장구조를 직접 설계하는 것이 응용에 따라서는 더 효율적으로 저장구조를 설계할 수 있다.

본 논문에서는 관계형 데이터베이스에서 DTD를 기반으로한 XML의 효율적인 문서 저장 구조를 설계한다. 제안된 구조는 사용자가 문서 스키마를 관계형 스키마로 사상시킬 때 그 저장구조에 따르는 질의처리가 효율적으로 이루어질 수 있도록 한다.

2. 관련연구

관계형 모델을 사용하여 XML 데이터를 저장하는 방법은 다음과 같이 세가지로 나눌수 있다.

첫째, DTD와 같은 기존의 스키마에 의존하지 않고, 반구조적 데이터를 분석해서 관계형 데이터 모델로 변환하는 방법이 있다.[1] 이 시스템은 반구조적 데이터가 주어지면 데이터 마이닝 기술을 이용하여 관계형 모델로 자동으로 변환되어 저장된다. 둘째, DTD가 주어졌을 때, DTD를 바탕으로 하여 XML 엘리먼트를 관계 테이블로 사상시키는 방법이 있다.[4] 이 경우에는 XML 엘리먼트와 관계 테이블 사이의 사상이 자동적으로 이루어진다. 셋째, Oracle 8i처럼 사용자나 시스템 관리자가 XML 문서를 관계형 테이블로 사상하는 방법을 제공해 주어야 하는 경우도 있는데[5], 본 논문에서는 Oracle 8i와 같은 시스템을 사용할 때, DTD를 기반으로 하며, XML 문서 질의처리의 효율성을 높일수 있는 사상 구조를 제시한다.

XML DTD를 관계형 테이블로 사상시키는 방법으로서 첫째, DTD에서 각 엘리먼트를 하나의 테이블로 사상하고, 엘리먼트의 애트리뷰트를 테이블의 필드로 두는 방법이 있다. 엘리먼트와 테이블간의 직접적인 사상은 문서의 과도한 단편화를 초래하는 단점을 가지기도 한다.[4] 둘째, 엘리먼트의 많은 자손들을 가능한 한 테이블의 필드로 둠으로서 단편화 문제를 해결하는 방법이 있다[4]. 이 방법은 DTD에 나와 있는 모든 엘리먼트에 대해서 테이블이 만들어지고, 중복된 데이터 값이 저장되어진다. 셋째, DTD 그래프에서 in-degree가 없는 엘리먼트만을 하나의 테이블로 사상하고, 그 테이블의 필드는 엘리먼트의 자손들로 둠으로서 엘리먼트와 테이블이 1:1로 사상되는 결점을 보완하는 방법이 있다.[4] 이러한 방법들의 효율성을 검증하기 위해서는 많은 실험이 필요하다.

3. XML 문서 저장 구조

3.1 저장구조 설계

우선, 엘리먼트의 성격을 구별하기 위하여, 엘리먼트를 complex 엘리먼트와 atomic 엘리먼트로 나눈다. complex 엘리먼트는 자식으로 엘리먼트를 가지는 엘리먼트를 의미하여, atomic 엘리먼트는 자식으로 #PCDATA나 EMPTY 속성을 가지는 엘리먼트를 의미한다.

테이블을 생성하는 기준은 형제관계에 있는 atomic 엘리먼트만을 필드로 가지는 하나의 테이블을 생성하고, 이런 필드를 포함하는 테이블의 이름은 그 atomic 엘리먼트들을 자식으로 가지는 부모 엘리먼트명을 두기로 한다. 이러한 방법은 엘리먼트와 테이블을 1:1로 사상을 하는 방식과 비교해 볼 때 테이블의 대량 생산을 방지할 수 있다. 또한 값을 가지지 않는 엘리먼트에 의해서 생겨난 테이블로 인해서 생길수 있는 대량의 널값을 방지할 수 있다. 그리고, 테이블에 doc_id 라는 필드를 둠으로서, 하나의 문서가 나누어져 저장이 되더라도 서로의 연관관계를 명확히 할 수 있다. 예를 들어 그림 1과 같은 DTD에 대하여 생성된 테이블의 모습이 그림 2에 나와있다. DTD 예제 1에서 보여 주듯이 엘리먼트 name, age, gender 는 #PCDATA 값을 가지고 있는 형제관계의 atomic 엘리먼트들이다. 이러한 엘리먼트를 필드로 두고, 이 세 엘리먼트의 부모가 되는 genera이 테이블의 이름이 된다.

```
<!ELEMENT patient (general)>
<!ELEMENT general (name, age, gender)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT gender (#PCDATA)>
<!ELEMENT age (#PCDATA)>
```

그림 1. DTD 예제 1

general

id	name	age	gender	doc_id

그림 2. DTD 예제 1에 대한 테이블 구조

XML 문서 저장시 엘리먼트와 애트리뷰트간의 구별없이 테이블상에 저장하게 되면, 문서 복구시 애트리뷰트가 엘리먼트가 되어서 문서의 구조가 다르게 나올 수 있다. 따라서, 엘리먼트와 애트리뷰트간의 구별이 가능하게 하기 위하여, 필드상에 표시를 두어 문서 복구시 정확한 정보를 추출하게 한다. XML 문서에서 발생되는 셋값 처리에 있어서, 같은 형제관계에 있는 atomic 엘리먼트들이라 할지라도 셋값이 있는 엘리먼트는 별개의 테이블로 처리하여 셋값을 가지지 않는 다른 엘리먼트의 중복된 값을 허용하지 않는다.

그림 3에서 보여주고 있는 DTD를 살펴보면, 두 개의 atomic 엘리먼트가 보인다. vdrl 엘리먼트와는 달리 polyp이라는 엘리먼트는 "+"을 셋값으로 가지고 있다. 따라서, 테이블의 생성모습은 그림 4, 5와 같이 combine 테이블과 polyp 테이블로 나누어져 있다. 결과적으로 *polyp* 셋값에 의한 *vdrl* 값의 중복된 값을 막을 수 있다.

```
<!ELEMENT patient (preexam)>
<!ELEMENT preexam (combine)>
<!ELEMENT combine (vdrl, polyp+)>
<!ELEMENT vdrl EMPTY>
<!ATTLIST  vdrl vdrlvalue (Y|N) "N">
<!ELEMENT polyp EMPTY>
<!ATTLIST  polyp id ID #IMPLIED
                 tmiste (A|M|C|AMC) #IMPLIED>
```

그림 3. DTD 예제 2

combine

id	vdrl.vdrlvalue.ATT	doc_id

그림 4. DTD 예제 2에 대한 테이블 구조

polyp

id	polyp.id.ATT	polyp.tmsite.ATT	doc_id

그림 5. DTD 예제 2에 대한 테이블 구조

XML 문서에서 발생되는 또 하나의 문제는 엘리먼트의 순환이다. 엘리먼트의 순환이라는 것은 자식 엘리먼트가 또 다시 자신의 부모 엘리먼트를 자식으로서 가지는 경우를 말한다. 트리 형태로 표현하면, 사이클이 생기는 경우를 의미한다. 이러한 순환은 문서의 구조를 바꾸어 놓기 때문에, 별도의 테이블을 생성한다. 순환되는 모양을 살펴볼 때, 반복적으로 나타나는 구조가 있다. 그 구조를 하나의 테이블로 생성한다. 이러한 순환으로 인해서 생성된 테이블에는 다른 테이블과는 달리 "parentID" 라는 별도의 필드를 둔다. 그리하여, 순환된 엘리먼트가 어느 엘리먼트에서 파생된 엘리먼트인지 문서의 구조관계를 명확히 한다.

그림 6에 나타난 DTD의 예제를 보면, 마지막 rec 엘리먼트가 자신의 부모 엘리먼트인 opinfo를 다시 자식으로 갖는 순환구조가 발생할 수 있음을 보여준다. 이 DTD 구조에서 반복되는 구조는 rec 아래에 opinfo가 오는 구조이다. 따라서, 그 반복구조는 그림 8과 같이 별도의 테이블을 생성한다. 그리고, 순환에 의해서 발생하는 테이블에는 별도의 parent_id 필드가 추가됨을 알수 있다.

```
<!ELEMENT patient (opinfo)>
<!ELEMENT opinfo (opdate, opname, rec)>
<!ELEMENT opdate (#PCDATA)>
<!ELEMENT opname (#PCDATA)>
<!ELEMENT rec(opinfo*|(#PCDATA))>
```

그림 6. DTD 예제 3

opinfo

id	opdate	opname	rec	doc_id

그림 7. DTD 예제 3에 대한 테이블 구조 (순환되지 않은 구조)

rec.opinfo

id	opdate	opname	rec	parent_id	doc_id

그림 8. DTD 예제 3에 대한 테이블 구조 (순환된 구조)

마지막으로 문서의 내용과 관련있는 위의 테이블과는 별도로 DTD 정보 및 테이블의 정보를 담고 있는 테이블이 있다. 이 테이블로 인해서 데이터베이스에 저장되어 있는 문서를 그대로 복원가능하게 하고, 질의처리시 데이터베이스 접근횟수를 줄이고자 한다.

Elementinfo

id	elename	setvalue	attribute	property	parent

그림 9. Element 정보를 담고 있는 테이블

그림 9에서 제시된 테이블에서 elename 필드는 해당 엘리먼트의 이름을, setvalue 필드는 해당 엘리먼트가 셋값을 가졌을 경우, 그 값의 형태를 표시한다. attribute 필드는 해당 엘리먼트가 애트리뷰트를 가졌는지에 대한 유무를 말하여, property 필드는 엘리먼트의 성격, 즉 complex 엘리먼트인지, atomic 엘리먼트인지를 기입하여, parent 필드는 해당 엘리먼트의 부모 엘리먼트를 가르킨다.

Attributeinfo

id	attname	elename

그림 10. Attribute 정보를 담고 있는 테이블

그림 10에서 제시된 테이블은 attribute 정보를 담고 있는 테이블로서, attribute 이름과 해당 엘리먼트 이름을 쌍으로 구성되어 있다.

tableinfo

id	elename	is_field	is_table

그림 11. table 정보를 담고 있는 테이블

그림 11에서 제시된 테이블은 앞에서 생성된 테이블에 대한 정보를 담고 있는 테이블로서 해당 엘리먼트가 필드로 사용되고 있는지, 테이블 이름으로 사용되고 있는지에 대한 boolean 형태로 제공되고 있다.

3.2 설계 예

여기에서는 앞에서 제시한 설계방향을 토대로 실제적인 문서가 저장되는 예를 살펴본다. 그림 12와 같은 XML 문서가 있을 때 이 문서가 관계형 테이블상에 저장된 내용은 그림 13과 같다..

```
<?xml version-"1.0"?>
<patient>
   <general>
      <name>park</name>
      <age>26</age>
      <gender>f</gender>
   </general>
   <preexam>
      <combine>
         <vdrl vdrlvalue-"n"/>
         <polyp id-"1" tmsite-"A"/>
         <polyp id-"2" tmsite-"ACM"/>
      </combine>
   </preexam>
   <opinfo>
      <opdate>761028</opdate>
      <opname>stomach</opname>
      <rec>
         <opinfo>
            <opdate>760702</opdate>
            <opname>river</opname>
            <rec>no</rec>
         </opinfo>
      </rec>
   </opinfo>
</patient>
```

그림 12. XML 문서(문서명 : x1.xml)

general

id	name	age	gender	doc_id
g1	park	26	f	x1.xml

combine

id	vdrl.vdrlvalue.ATT	doc_id
c1	n	x1.xml

polyp

id	polyp.id.ATT	polyp.tmsite.ATT	doc_id
p1	1	A	x1.xml
p2	2	AMC	x1.xml

opinfo

id	opdate	opname	rec	doc_id
o1	761028	stomach		x1.xml

rec.opinfo

id	opdate	opname	rec	parent_id	doc_id
r1	760702	river	no	o1	x1.xml

Elementinfo

id	elename	setvalue	attribute	property	parent
e1	patient		false	complex	
...	...	...	...	...	...

Attributeinfo

id	attname	elename
a1	vdrlvalue	vdrl
a2	id	polyp
a3	tmsite	polyp

tableinfo

id	elename	is_field	is_table
t1	patient	false	false
t2	general	false	true
...	...	...	...

그림 13. XML 문서의 저장구조

4. 구현

본 논문에서는, 데이터베이스에 저장된 정보를 추출하기 위한 질의어로서 XQL을 사용하였다. XQL은 1998년 w3c에서 제안한 XML 문서를 위해서 특별히 설계된 질의어이다[6]. 이 XQL을 사용하여 DB에 있는 정보를 추출하기 위해, SQL로 변환하는 알고리즘을 설계하였다. 현재, 엘리먼트, 애트리뷰트, 필터링과 같은 XQL의 일부기능이 구현되어 있다. 그림 14는 구현된 결과의 일부를 보인 것이다.

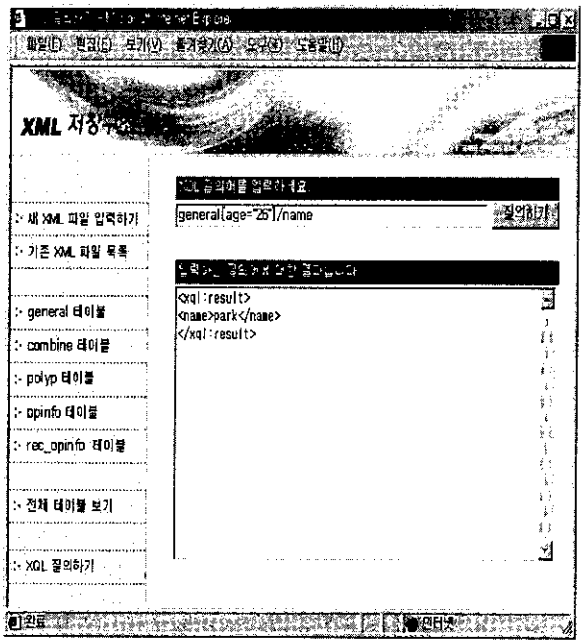

그림 14. XQL 질의와 검색결과

5. 결론

본 논문에서는 관계형 모델을 사용하여 DTD기반의 XML 데이터를 저장하는 구조를 설계하였다. 관계형 모델을 이용하여 XML 데이터를 저장하는 여러가지 메커니즘들이 존재하지만, 본 논문에서는 사용자나 관리자가 XML 문서를 관계형 테이블로 직접 사상시킬때의 구조를 설계하였다. 이 방법은 질의처리의 효율성이 사용자의 데이터베이스 설계 경험에 크게 의존하는 단점이 있으나, 문서의 구조를 설계할 때, 복잡한 DTD를 단순한 형태로 변형시킬 필요가 없기 때문에 데이터베이스에 저장된 XML 문서로부터 원래의 XML 문서로 그대로 복원하기가 쉽고, 문서의 구조를 파악하기 위하여 시스템내에 DTD 그래프를 생성시킬 필요가 없으며, 재귀적 종속관계에 있는 엘리먼트를 찾는 과정에 대한 처리가 필요없다.

이에 본 논문에서 제시된 사상구조는 현재 보편화되어 있는 관계형 데이터베이스 시스템하에서 XML 문서를 효과적으로 관리하는데 적용될 수 있다.

참고문헌

[1] Alin Deutsch, M.F ernandex,D.Suciu, "Storing Semi-structured Data with STORED," Proceedings of ACM SIGMOD Conference. Philadelphia, Pennsylvania, May 1999

[2] D.Florescu and D.Kossmann, "Storing and Querying XML Data using as RDBMS," Bulletin of the Technical Committee on Data Engineering, Vol. 22, No.3, 1999

[3] J.McHugh, S.Abiteboul, R.Goldman, D.Quass, and J.Widom, "Lore : A database management system for semistructured data," Technical report, Stanford University Database Group, Februdary 1997

[4] J.Shanmugasundaram, K.Tufte, C.Zhang, G.He, D.J.DeWitt, and J.F.Naughon, "Relationa Database for Queyring XML Documents : Limitations and Opportunities," Proc. of 25th Int'l Conf. on VLDB, Edinburgh, Scotland, UK, 1999

[5] Oracle Corporation, "XML Support in Oracle 8 and beyond," Technical white paper, "http://www.oracle.com/xml/documents

[6] XML Query Language(XQL), "http://www.w3.org/TandS/QL/QL98/pp/xql.html"