XML 구성요소의 릴레이션으로의 변환

신병주, 진 민
경남대학교 컴퓨터공학과
e-mail:{challenger,mjin}@hawk.com.kyungnam.ac.kr

Transferring XML Documents to Relational Scheme

Byung-Joo Shin, Min Jin
Dept of Computer Engineering, Kyungnam University

요약

XML 문서의 사용이 급속도로 증가함에 따라 대용량의 XML 문서를 저장, 관리하는 기술이 요구되고 있다. XML 문서를 저장, 관리방법으로 RDBMS가 현실적으로 가장 효과적인 방법이다. 그러나, XML의 구성요소와 RDBMS의 구성요소간의 차이로 인해 XML 문서를 RDBMS에 저장하기 위해서는 특별한 저장방법이 제공되어져야 한다. 따라서, 본 논문은 이와 같은 XML과 RDBMS 구성요소간의 불일치에서 오는 문제점들을 해결하고 효율적인 질의처리가 가능하도록 XML의 각 구성요소들에 대한 저장방법을 제시한다.

1. 서론

인터넷의 기반 기술로 확고히 자리잡아 가고 있는 XML은 코딩의 형태가 간단하고 습득이 용이하고 우수한 확장성과 융통성을 제공한다. 즉, XML은 HTML과 같은 고정된 태그가 아닌 문서의 구조와 의미에 관한 정보가 포함되어 있는 사용자 정의의 태그를 사용하여 확장성과 유연성 등의 강력한 장점을 가지고 있다.

현재 XML은 강력한 장점을 제공하면서 인터넷을 기반으로 하는 모든 영역으로 확산, 적용되고 있다. XML 문서의 양이 많아져 이것을 저장, 관리할 XML 저장관리 기술이 절실히 요구되고 있다. 대용량의 XML 문서를 저장, 검색, 관리하기 위해서는 XML의 특성에 맞는 저장관리 시스템이 필요하다.

RDBMS를 이용하여 XML을 저장하는 방식은 현재 RDBMS 시장이 다른 DBMS에 비해 넓고 사용자 수도 많기 때문에 쉽게 상용화할 수 있다는 장점이 있다. 또한, RDBMS는 기존의 일반적인 데이터와 XML과 같은 반구조적인 데이터가 공존할 수 있다는 장점이 있고, 대용량의 XML 문서에 대한 복잡한 질의처리 능력이 기존의 다른 DBMS보다 우수하다.

현실적으로 XML을 저장하기에 가장 효과적이라고 할 수 있는 RDBMS을 사용하여 XML 데이터를 저장하는 방법에는 XML 문서를 저장하는 방법을 사용자나 시스템 관리자가 결정하는 방법과 기존의 XML 스키마에 의존하지 않고 반구조적인 XML 데이터를 분석하여 저장하는 방법이 있다. 그리고 XML 스키마를 기반으로 저장구조를 설계하는 방법도 있다. 특히, 이 중에서 XML 스키마인 DTD로부터 XML 문서의 저장방법을 추론하는 방법은 현실적으로 중요한 위치를 차지하고 있다[3].

따라서, 본 논문은 DTD 기반의 XML 문서를 RDBMS에 효율적으로 저장하기 위한 방법을 제안하고자 한다. 특히, XML의 장점을 유지하면서 XML 문서의 각 구성요소들을 RDBMS에 저장하고 효율적인 질의처리가 가능한 저장구조의 설계 방법을 제안한다.

2. 관련연구

XML의 구성요소들로는 element, attribute, entity 등이 있다. element는 element-only element, text-only element, empty element, mixed element,

*본 논문은 과학기술부와 한국과학재단으로부터 지정받은 경남대학교 인간역 폐자원 및 환경연구센터(CRERC)의 지원에 의하여 연구되었음.

any element 등의 형식이 있다. element-only element는 단지 다른 element를 하위 구성요소로 가지는 element이다. text-only element는 #PCDATA로 선언되어져 text string을 가지는 element이다. empty element는 하위 구성요소가 없다. 이 empty element는 element에 추가적인 정보가 data point 레벨의 데이터일 경우에 attribute들을 이용해서 표현한다. 그리고 mixed element는 하위 구성요소로서 0개 이상의 선언되어진 element들이 어떤 순서나 위치에 관계 없이 나타난다. 마지막으로 any element는 특정한 구조정보가 없다고 볼 수 있다. 즉, 하위 구성요소로서 제한없이 모든 형태의 구성요소가 올 수가 있다. 따라서, mixed element와 any element는 다른 element들에 비해 저장시 복잡한 처리과정이 필요하게 된다[1][2].

XML의 또 다른 구성요소인 attribute는 element에 대한 정보를 표현하는 것이라 볼 수 있다. attribute list의 선언은 정의되어진 element에 대해서 범위와 타입을 제한한다. 허용가능한 attribute들의 type들을 보면 문자열을 포함하는 CDATA, 나열된 값 중에서 하나를 선택할 수 있는 enumerated list attribute, attribute 값으로 객체를 삽입하기 위한 entity(ies) attribute, element의 유일한 식별자 기능을 담당하는 ID attribute, 문서 내부에서 이전에 선언된 유일한 ID를 참조하는 IDREF attribute, CDATA와 유사하지만 공백을 허용하지 않는다는 제한이 있는 NMTOKEN(S) 등이 있다[1][2].

이외에도, 특정 객체들을 XML 문서에 삽입하기 위한 ENTITY 등의 구성요소들이 존재한다.

이와 같이, 다양한 XML의 구성요소들을 RDBMS에 저장하기 위해서는 약간의 처리과정이 필요하다. 왜냐하면, XML의 구성요소들의 구조와 RDBMS의 구조가 일치하지 않기 때문이다. 즉, XML의 element, attribute의 구조와 관계 스키마의 릴레이션, 속성의 구조 관계가 항상 일치하지 않는다. 따라서, 데이터의 단편화 문제, 질의처리의 효율성 등의 문제를 해결하는 방법이 제공되어져야 한다[4][5].

3. XML 구성요소의 저장을 위한 저장구조

3.1 Element

3.1.1 Element-Only Element

Element-Only Element는 하위 구성요소로 다른 element를 가지기 때문에 관계 스키마로 사상시 릴레이션으로 사상되어져야 한다. 이때, 하위 element들과의 관계는 외래키를 이용하여 표현하도록 한다. 그런데, <그림 1>의 예에서 Name element는 Professor element와 Student element의 복수 개의 상위 Element-Only Element로부터 공유되어지는 경우가 발생한다. 이와 같은 경우에는 상위 element의 종류를 가리키는 ParentCode 속성을 생성한다.

```
<!ELEMENT School (Professor*, Student*)>
<!ELEMENT Professor (PNum, Name)>
<!ELEMENT PNum (#PCDATA)>
<!ELEMENT Student (SNum, Name)>
<!ELEMENT SNum (#PCDATA)>
<!ELEMENT Name (#PCDATA)>
```

Name Table

NameKey	ParentCode	Name
1	Professor	Min Jin
2	Student	Byung-Joo Shin

<그림 1> 복수 개의 상위 element 저장 예

3.1.2 Text-Only Element

Text-Only Element는 하위 구성요소로 text 스트링을 갖는다. 따라서, 관계 스키마 사상시 릴레이션의 속성으로 사상되어지는 것이 효율적이다. 만약, <그림 2>의 Class element와 같이 *, + 등의 연산자에 의해 다중값을 갖는 Text-Only Element인 경우에는 별도의 릴레이션을 생성하여 외래키를 이용하여 상위의 element와 연결하도록 한다.

```
<!ELEMENT Student (SNum, Name, Class*)>
<!ELEMENT SNum (#PCDATA)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Class (#PCDATA)>
```

Student Table

StudentKey	SNum	Name
1	1993174030	Byung-Joo Shin
2	1995174001	Dong-Jin Lee

Class Table

StudentKey	Class
1	Data Structure
1	System Programming
2	Data Structure
2	Web Programming

<그림 2> 다중값을 갖는 Text-Only Element 저장 예

3.1.3 Empty Element

Empty Element는 하위 구성요소가 존재하지 않고 Element에 대한 추가정보가 ATTLIST를 통해 제공되어진다. 따라서, Empty Element는 릴레이션으로 사상하고 Element에 대한 추가정보인 ATTLIST의 데이터들은 릴레이션의 속성으로 사상한다.

3.1.4 Mixed Element와 Any Element

mixed element와 any element는 하위 element로 text 데이터(#PCDATA)나 여러 형태의 element들이 어떠한 순서에 관계없이 존재할 수 있으므로 관계 데이터베이스에 저장하기 위해서는 별도의 처리과정이 필요하게 된다.

```
<!ELEMENT p (#PCDATA | a | br | span | bdo | object | img | map | tt | i |
b | big | small | em | strong | dfn | code |.....)*>

<p>In the mixed content model, elements may contain <b>zero or more
instances of a list of elements, </b>in any order, <i>along with any amount of
text in any position</i>. This is an example of the mixed content model.</p>
```

<그림 3> Mixed Element의 예

<그림 3>은 Mixed element의 예이다. 위의 예에서 제시된 p element를 저장하기 위해 p 테이블을 생성한다. 그리고 p의 하위 element들에 대한 정보를 저장하는 pSubelement 테이블에는 p 테이블과 연결을 위한 외래키의 역할을 담당하는 pKey 속성, 순서 정보를 표현하는 sequence 속성, 하위 element의 종류를 표현하는 TableName 속성, 각 하위 element table에서의 특정 row를 가리키는 LookupKey 속성 등으로 구성된다. 그리고 하위 element들을 저장하는 테이블 이름의 유효성 체크를 위한 별도의 테이블이 필요하다. <그림 4>는 <그림 3>의 mixed element를 저장한 예이다. <그림 4>의 예에서는 p의 하위 element인 b, i element를 text-only element로 가정하였다. <그림 4>의 pSubelement 테이블의 TableName 속성은 하위 element의 명을 저장함으로써, 하위 element와 연결할 수 있도록 한다. 그 중에서 TextContent는 하위 element가 #PCDATA로 선언되어져 text의 값을 가질 때를 나타낸다.

p Table

pKey
1

pSbuelement Table

pKey	TableName	LookupKey	Sequence
1	TextContent	1	1
1	b	1	2
1	TextContent	2	3
1	i	1	4
1	TextContent	3	5

Valid Table

Key	TableName
1	TextContent
2	a
:	:
11	b
:	

TextContent Table

TextContentKey	TextContent
1	In the mixed content model, elements may contain
2	in any order,
3	This is an example of the mixed content model.

b Table

b_Key	bContent
1	in any order,

i Table

i_Key	iContent
1	along with any amount of text in any position

<그림 4> Mixed Element의 저장 예

Any Element의 저장은 Mixed Element의 저장방법과 같은 방법으로 가능해진다.

3.3 Attribute

3.3.1 CDATA

CDATA type의 attribute는 파싱되지 않는 문자 데이터가 온다. CDATA attribute는 #REQUIRED, #IMPLIED, #FIXED 등의 디폴트 값들을 가질 수가 있다. #REQUIRED를 디폴트 값으로 가지는 CDATA attribute는 관계데이터베이스에서 NULL을 허용하지 않는다. 이와는 달리 #IMPLIED를 디폴트 값으로 가지는 CDATA attribute는 NULL을 허용하는 값을 가지게 된다. 한편, #FIXED를 가지는 CDATA attribute는 attribute를 정의한 후 XML 문서 내에서 attribute 값을 어떤 다른 값으로 변경을 허용하지 않는다. 따라서, #FIXED를 가진 CDATA attribute를 관계 데이터베이스에 저장할 때는 그 attribute가 사상되어진 속성에는 항상 특정 값이 저장되어져야 한다.

3.3.2 열거형

열거형 attribute는 나열된 값 중에서 하나만이 선택된다. 이 열거형 attribute는 관계 데이터베이스에서 lookup 테이블을 통해서 모델링한다. 그러나 열거형 attribute를 갖는 element의 명은 다르지만 attribute의 명이 같음으로써 의미가 다른 두 개의 열거형 attribute들의 값들이 한 테이블에 저장되는 문제가 발생하게 된다. 즉, <그림 5>의 예에서 City라는 열거형 attribute는 두 개의 다른 empty element(Korea element, U.S.A. element)의 정보를 한 테이블에서 저장하고 있다. 이와 같은 경우를 위해 lookup 테이블 생성시 element명을 붙인 lookup 테이블을 별도로 생성하거나 아래와 같이 element를 지시하는 별도의 속성을 생성한다.

```
<!ELEMENT Korea EMPTY>
<!ATTLIST Korea
   City (Seoul | Pusan | Taegu) #REQUIRED>
<!ELEMENT U.S.A. EMPTY>
<!ATTLIST U.S.A.
   City (New York | L.A | Chicago ) #REQUIRED>
```

CityLookup Table

TableName	CityKey	Value
Korea	1	Seoul
Korea	2	Pusan
Korea	3	Taegu
U.S.A.	1	New York
U.S.A.	2	L.A
U.S.A.	3	Chicago

<그림 5> 열거형 attribute의 저장 예

3.3.3 ID and IDREF(S)

ID type의 attribute의 ID 값은 XML 문서의 모든 element에 대해서 유일한 식별자를 가지게 된다. IDREF는 문서 내부에서 이전에 선언된 유일한 ID를 가리킨다. IDREFS는 동시에 복수 개의 ID를 참조하는 것이 가능하다.

따라서, 문서 내부에서 특정한 ID 값만을 IDREF로 선언되어진 attribute가 참조한다면 primary key, ID 속성, IDREF의 속성의 필드를 갖는 테이블을 생성하면 된다. 그러나, IDREF는 XML 문서에서 특정한 ID가 아닌 복수 개의 ID로 선언된 값들을 참조할 수도 있다. 또한 IDREFS는 ID와 M:N의 관계를 갖는다. 따라서, 이런 문제를 해결하기 위하여 <그림 6>의 예처럼 primary key, ID 속성값, IDREF(S)의 속성값을 나타내는 속성 외에도 참조하는 테이블명을 저장하는 속성을 추가로 생성하여 그 값을 저장한다.

```
<!ELEMENT CPU EMPTY>
<!ATTLIST CPU
   CPUID ID #REQUIRED>
<!ELEMENT Monitor EMPTY>
<!ATTLIST Monitor
   MonitorID ID #REQUIRED>
<!ELEMENT Computer EMPTY>
<!ATTLIST Computer
   ComputerID ID #REQUIRED
   ComputerAccessoryIDREFS IDREFS #REQUIRED>

<CPU CPUID="Intel100"  />
<Monitor MonitorID="LGIBM200"  />
<Computer ComputerID="Computer1"
                        ComputerAccessoryIDREFS="Intel100 LGIBM200"
<Item ItemID="Computer2"   ComputerAccessoryIDREFS="Intel100"
```

InvoiceIDREF Table

Key	ID	TableName	IDREF
1	computer1	CPU	Intel100
2	computer1	Monitor	LGIBM200
3	computer2	CPU	Intel100

<그림 6> ID와 IDREFS의 저장 예

3.3.4 NMTOKEN(S)

NMTOKEN은 공백 글자를 허용하지 않는다는 점을 제외하고는 CDATA와 유사하다. 그리고 NMTOKEN은 하나의 NMTOKEN 값만을 허용하는 것에 반해, NMTOKENS는 한 개 이상의 NMTOKEN 값이 올 수 있다. 그러므로, NMTOKEN은 릴레이션의 속성으로 사상하고, NMTOKENS일 경우에는 별도의 테이블을 생성하여 NMTOKEN 값들의 순서, NMTOKEN 값들을 저장하고 외래키로 상위 element와 연결한다.

4. 결론

본 논문은 XML 문서의 RDBMS 저장시 XML의 구성요소들의 구조와 RDBMS의 구성요소간 구조의 차이에서 오는 문제점을 해결하고 효율적인 질의처리를 위한 저장구조를 설계하였다. 즉, XML의 element, attribute의 구조와 관계 데이터베이스의 릴레이션, 속성의 구조가 일치하지 않기 때문에 XML의 구성요소를 관계 데이터베이스로의 저장시 사상방법에 대한 특별한 처리과정이 필요하다. 그리고, 다중값을 갖는 element의 저장과 각 구성요소들간의 다대다 관계 등의 표현을 위한 추가적인 저장방법을 제안하였다. 또한, 데이터의 중복과 단편화 등의 문제를 해결하여 효율적인 질의의 처리가 가능한 저장구조를 설계하였다.

참고문헌

[1] F. Boumphrey, O. Direnzo, J. Duckett, J. Graf, P. Houle, D. Hollander, T. Jenkins, P. Jones, A. Kingsley-Hughes, K. Kingsley-Hughes, C. McQueen, S. Mohr, *Professional XML Applications*, Wrox Press, 1999

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

[3] D. Florescu, D. Kossmann, "Storing and Querying XML Data using an RDBMS", Data Engineering Bulletin, Vol. 22, No. 3, 1999

[4] D. Kroenke, *Database Processing*, Prentice Hall, 2000

[5] K. Williams, M. Brundage, P. Dengler, J. Gabriel, A. Hoskinson, M. Kay, T. Maxwell, M. Ochoa, J. Papa, M. Vanmane, *Professional XML Databases*, Wrox Press, 2000