XML 문서의 관계 데이터베이스 저장

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

Storing XML Documents in Relational Databases

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

요약

XML 문서의 저장 및 관리 방법으로 관계형 데이터베이스가 많이 사용되고 있다. 그러나 XML과 관계 데이터베이스의 구조상의 불일치로 인해 데이터 손실, 질의처리의 효율성 저하 등과 같은 문제가 발생한다. element의 순환, 다중값을 갖는 attribute들의 처리 등을 위한 별도의 처리방법이 필요하다. 따라서, 본 논문은 효율적인 저장, 관리 및 질의 처리를 고려하여 DTD 기반의 XML 문서를 관계 데이터베이스에 저장하는 방법을 제시한다.

1. 서론

XML은 의미있는 사용자 정의 태그를 사용할 수 있고 문서에 대한 구조정보를 제공할 뿐만 아니라, XML 태그는 데이터를 해석하는데 사용할 수 있기 때문에 인터넷 상에서 데이터를 표현하고 교환하는 새로운 표준으로 그 중요성이 증대하고 있다.

XML 데이터는 파일시스템, RDBMS, OODBMS, 반구조적 데이터를 위한 고유 시스템 등에 저장될 수 있다. 파일시스템은 XML 데이터의 저장, 문서 전체에 대한 검색은 용이하나 문서의 갱신 등과 같은 특정한 질의의 처리에 대한 지원이 부족하다. OODBMS는 이상적으로는 객체 기반에 구조를 둔 XML 문서를 저장하기에 가장 적합한 저장소이다. 그러나, 현실적으로 대용량의 데이터베이스에 대한 복잡한 질의를 처리하기에는 아직 어려움이 많다. 그리고 반구조적 데이터를 위한 고유 시스템은 데이터 처리를 위한 복잡한 과정이 필요하고 과다한 저장공간의 낭비 등 아직 XML 문서를 저장, 관리하기에는 미비한 수준이다. 이에 반해, RDBMS는 일반적인 데이터 뿐만 아니라 반구조적 데이터가 공존할 수 있고 대용량 데이터베이스에 대한 복잡한 질의의 처리에 효율적이다. 따라서, 현실적으로 RDBMS는 XML 문서를 저장, 관리하기에는 가장 효율적인 저장 방법이라고 할 수 있다[3].

그러나, 객체 기반의 XML의 구조와 관계 데이터베이스의 구조는 일치하지 않다. 즉, XML의 element, attribute와 관계 스키마의 릴레이션, 속성이 항상 일치하는 것이 아니다. 그럼으로써, 데이터의 단편화 문제, 다중값을 가지는 (Set-valued) attribute, element의 순환(recursion) 등의 처리 문제, 질의처리의 효율 저하 등의 문제가 발생한다.

따라서, 본 논문은 기존의 관련 연구를 바탕으로 XML 문서의 관계 데이터베이스 저장시 구조상의 불일치로 발생하는 문제들을 해결하고 효율적인 질의처리를 고려한 사상방법을 제시하고자 한다.

2. 관련연구

관계 스키마에 DTD 기반 XML 문서를 사상하는 가장 기본적인 방법은 element는 릴레이션으로, attribute는 릴레이션의 속성으로 사상하는 것이다. 그러나, 이와 같은 방법은 문서의 과도한 단편화 문제가 발생하게 된다. 따라서, 이런 문제를 해결하기 위한 방법으로 DTD 그래프와 element 그래프를 이용하여 특정 element들을 릴레이션의 속성으로 표현하는 방법이 제시되었다[4].

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

Basic Inlining 방법[4]은 element를 릴레이션으로, 각 element의 하위 element는 속성으로 inline한다. 그리고 다중값을 갖는 element와 순환구조를 갖는 element에 대해서는 별도의 릴레이션을 생성한다. 그러나 Basic Inlining 방법은 과도한 릴레이션의 생성과 하나의 element가 여러 릴레이션으로 나누어져 저장됨으로써 질의처리의 어려움이 발생한다.

Shared Inling 방법[4]은 DTD 그래프 상에서 in-degree의 값에 따라 element와 attribute를 사상하여 Basic Inling 방법의 단점을 보완한다. 즉, in-degree가 0이거나 2 이상일 경우에는 릴레이션으로, 1일 경우에는 속성으로 사상한다. 그리고 다중값을 갖는 element와 element의 순환이 발생하는 경우에는 별도의 릴레이션을 생성하고 있다. 이와 같은 Shared Inling 방법은 Basic Inling 방법에서 과도한 릴레이션이 발생하는 단점을 해결하였으나 질의 처리시 조인 연산이 증가하게 되는 단점을 갖게 된다.

Hybrid Inling 방법[4]은 다중값을 갖는 element, element의 순환이 발생하지 않고 in-degree가 2이상인 element들을 속성으로 inline함으로써 Shared Inling 방법의 단점인 조인 연산의 횟수를 감소시킨다. 그러나 Hybrid Inling 방법은 특정 element들을 여러 릴레이션의 속성으로 inline함으로써 특정 질의의 처리시 여러 릴레이션에서 질의 처리된 값들을 합쳐야하는 복잡한 과정이 필요하게 된다.

XML 문서를 RDBMS에 저장하는 다른 방법으로는 XML 문서를 순서와 레이블이 있고 방향이 존재하는 그래프로 표현하여 각 element는 그래프에서 노드로, 하위 element와의 관계는 에지로 표현한다. 그리고, XML 문서의 value들은 그래프의 leaf로 표현한다. 그리하여 그래프 상의 에지와 Value가 저장되는 방법에 따라 저장방법을 제시하는데, 에지가 저장되는 방법에는 그래프의 모든 에지들을 하나의 테이블에 저장하는 에지 방법, 하나의 테이블에 같은 레이블을 가지는 에지들만을 그룹화시켜 저장하는 Binary 방법, Binary 방법으로 생긴 테이블들을 외부 조인한 것과 같은 Universal 방법 등이 있다. 그리고 Value가 저장되는 방법은 각 데이터형에 따라 별도의 테이블을 생성하여 값을 에지들과는 별도로 저장하는 방법과 에지들이 저장되는 테이블에 에지들과 같이 inline 저장되는 방법 등이 있다[3].

XML 구조는 객체 기반의 구조를 가지고 있기 때문에 XML 문서를 객체-트리의 구조라는 관점에서 저장할 수도 있다[1]. 즉, element들은 클래스로, attribute와 PCDATA 등은 속성 등으로 간주하여 클래스는 테이블로, 속성은 컬럼으로 저장하고 각 클래스간의 관계는 외래키를 이용해서 표현한다. 그리고 트리 구조 상에서 나타내어지는 계층 구조와 순서에 관한 정보도 저장함으로써 XML 문서의 구조도 저장할 수가 있다. 그러나 객체 기반의 XML 구조와 RDBMS의 구조 상에서 오는 불일치로 인하여 발생하는 문제들을 해결하기 위한 방법들이 별도로 마련되어져야 한다.

3. XML의 RDBMS 저장에 있어 문제점

XML 문서를 관계 데이터베이스로 사상하는 기본적인 방법은 XML DTD에서 각 element는 관계 스키마의 릴레이션으로, 각 element의 attribute는 릴레이션의 속성으로 사상하는 것이다. 그러나 문제는 이와 같이 XML 문서의 element, attribute의 구조와 관계 스키마의 릴레이션, 속성의 구조와 항상 일치하는 것이 아니라는 것이다. 즉, XML 문서의 구조와 관계 스키마의 구조를 일치시켜 저장하게 된다면 릴레이션의 대량 생산과 값을 가지지 않는 element에 의해 발생하는 저장 공간의 낭비를 초래하게 되는 것이다. 따라서, XML에서 element로 표현되어진 데이터를 관계 스키마에서는 속성으로 사상해야 하는 경우가 발생한다. 이와 같이, XML의 구조와 관계 스키마의 구조가 일치하지 않음으로써 발생하는 과도한 릴레이션의 생성, 저장 공간의 낭비, 일부 데이터의 손실 등과 같은 문제들을 해결할 수 있는 방법이 모색되어져야 하는 것이다.

그리고 XML 문서를 RDBMS에 저장하는 방법에 따라 질의에 대한 처리의 효율성에 영향을 미치게 된다. 즉, element를 관계 스키마의 속성으로 표현할 때, 그 방법에 따라 질의 처리시 과도한 조인 연산의 증가나 복잡한 질의문의 사용이 필요한 경우가 발생하는 것이다. 또한, 질의 처리의 효율성만을 고려한 일부 저장방법은 많은 널(Null) 값을 갖거나 데이터의 중복이 발생하여 저장 공간의 낭비가 심하게 발생하게 된다.

XML 문서는 구조와 내용을 분석하여 여러 릴레이션에 저장하게 된다. 또한, RDBMS는 다중값을 갖는 element와 element의 순환 등과 같은 문제를 해결하기 위한 방법을 제대로 제공하지 못한다. 따라서, 이러한 문제들을 해결하기 위해서는 데이터의 단편화를 초래할 수 밖에 없다. 이와 같은 데이터의 단편화로 인해 RDBMS에 저장되어진 XML 문서를

원래대로 재생산해 내기 위한 처리과정이 필요하다.

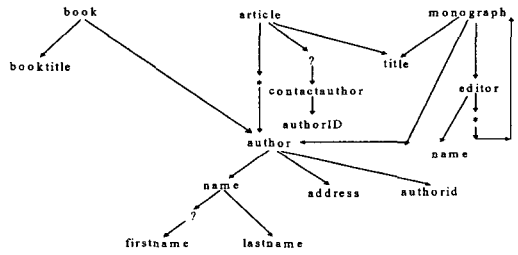
4. XML의 관계스키마로의 사상

4.1 릴레이션의 생성

```
<!ELEMENT book (booktitle, author)>
<!ELEMENT booktitle (#PCDATA)>
<!ELEMENT article (title, author*, contactauthor)>
<!ELEMENT contactauthor EMPTY>
<!ATTLIST contactauthor authorID IDREF IMPLIED>
<!ELEMENT monograph (title, author, editor)>
<!ELEMENT editor (monograph*)>
<!ATTLIST editor name CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (name, address)>
<!ATTLIST authorid ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address ANY>
```

<그림 1> DTD 예

모든 element를 릴레이션으로 사상함으로써 발생하는 대량의 릴레이션 생성 문제를 해결하기 위해 일부 element를 상위 element에 속성으로 표현(inline)한다. 또한, 다중값을 가지는 element, element의 순환 등과 같은 문제들을 해결하고 효율적인 질의 처리를 고려하여 저장방법을 결정한다.



<그림 2> DTD 그래프

XML 문서의 element들을 관계스키마의 릴레이션이나 속성으로 사상하는 기준은 먼저 DTD 그래프의 in-degree 값이다. <그림 2>의 DTD 그래프에서 book element처럼 in-degree 값이 0일 경우에는 상위 element가 존재하지 않으므로 릴레이션으로 사상한다. 그리고 atomic element(text-only element, empty element 등)는 일반적으로 in-degree가 1인 경우가 대부분인데, 이 때에는 element일지라도 상위 element에 포함시켜(inline) 관계스키마의 속성으로 사상하게 된다. 한편, 모든 종류의 element는 두 개 이상의 상위 element를 가질 수 있다. 즉, in-degree가 2이상일 경우에는 기본적인 사상방법은 별도의 릴레이션을 생성하는 것이다. 하지만, in-degree가 2이상일지라도 title element처럼 단말 노드라면 새로운 릴레이션을 생성하는 것보다는 상위 element에 속성으로 표현하는(inline) 방법이 더 효율적이다. 즉, in-degree가 2 이상일 경우에는 상위 element와 하위 element간의 상관관계의 분석을 통해 독립적인 관계이라면 릴레이션으로, 종속적인 관계라면 속성으로 표현(inline)하는 것이 좋다. 예를 들어, <그림 2>에서 author는 book과는 독립적인 개체로 간주되어 별도의 릴레이션을 생성하지만 name은 author와 editor의 성질을 기술하는 것으로 보아 상위 element의 속성으로 표현(inline)하는 것이 효과적이다.

그러나, 다중값을 갖는 attribute와 element의 순환이 발생하는 element는 in-degree의 값과는 무관하게 별도의 릴레이션을 생성한다.

4.2 릴레이션 생성시 추가되는 정보

각 릴레이션 생성시에는 키의 역할을 하는 ID 속성을 생성한다. 그리고 상위 element와의 관계의 표현을 위해 parentID라는 속성의 외래키를 생성한다. 그리고 복수개의 상위 element를 가지는 경우, 즉 in-degree가 2 이상일 경우에는 parentCode라는 속성을 통하여 상위 element를 구분하도록 한다.

특정 element를 릴레이션의 속성으로 사상했을 경우 이 element가 XML 문서의 루트가 되는 경우에 문제가 발생하게 된다. 데이터베이스 검색시 이러한 element에 대한 질의의 처리를 위해 isroot라는 속성을 두어 element가 속성으로 표현됨으로써 발생하는 문제를 해결한다[4]. 또한, XML 문서의 element들이 여러 릴레이션으로 분리되어 저장됨으로써 데이터간의 연관관계가 불명확해지게 된다. 따라서, doc_id라는 속성을 추가하여 연관관계를 표현하여 XML 문서를 원래대로 복원할 수 있도록 한다. <그림 3>은 <그림 2>의 DTD 그래프[4]를 통하여 생성되어진 릴레이션이다.

Book Table

id	BookTitle.isroot	BookTitle	doc_id

Article Table

id	ContactAuthor.isroot	Authorid	Title.isroot	Title	doc_id

Monograph Table

id	ParentID	ParentCode	Title.isroot	Title	Editor.isroot	Name	doc_id

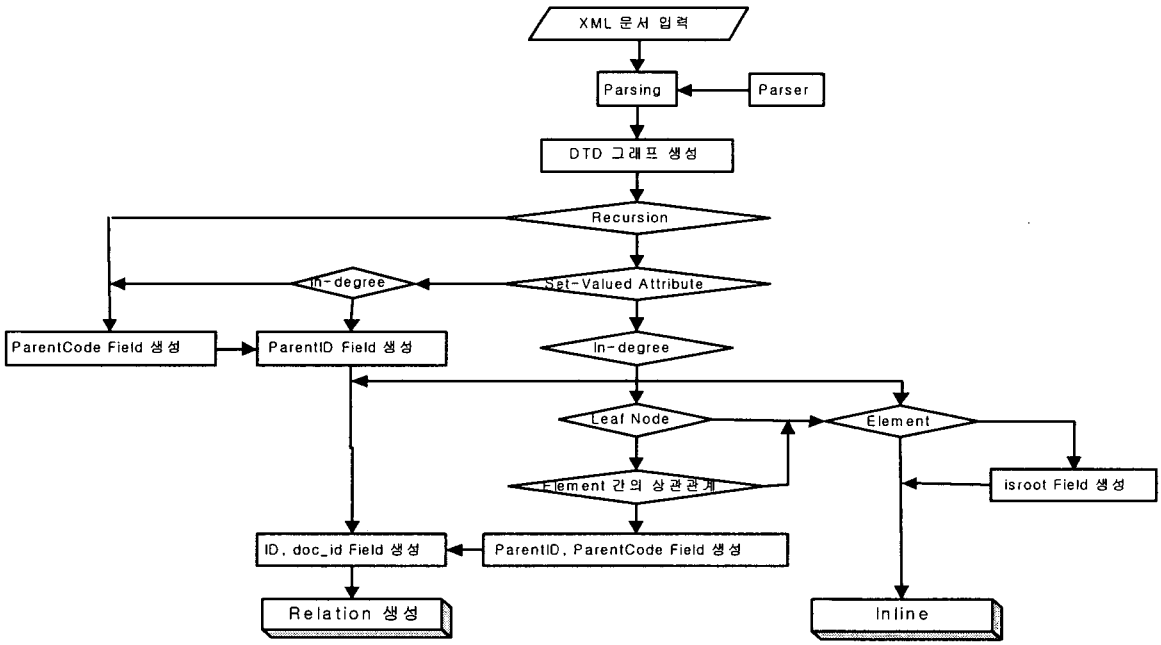
Author Table

id	ParentID	ParentCode	name.isroot	name	address.isroot	address	authorid	doc_id

<그림 3> 릴레이션의 생성

4.3 ElementInfo 테이블과 AttributeInfo 테이블

XML 문서가 여러 릴레이션에 분리되어 저장되어 XML 문서가 가지는 구조 정보가 손실된다. 따라서, XML 문서의 구조 정보를 저장하여 원래의 XML



<그림 4> 릴레이션 생성 알고리즘

문서의 복원이 가능하게 하고 질의 처리시 데이터베이스에 대한 접근 횟수를 감소시키기 위하여 DTD와 각 테이블에 대한 정보를 저장하는 ElementInfo 테이블과 AttributeInfo 테이블을 생성한다.

ElementInfo 테이블

ID	EleName	Operator	Attribute	Property	Parent	Mapping	doc_id
1	book	null	false	complex	null	entity	ex.xml
2	booktitle	null	false	aotmic	book	attribute	ex.xml
3	author	null	true	complex	book	entity	ex.xml
4	article	null	false	complex	null	entity	ex.xml
5	title	null	false	atomic	article	attribute	ex.xml
6	author	*	true	complex	article	entity	ex.xml
:	:	:	:	:	:	:	:

AttributeInfo 테이블

ID	ElementName	AttributeName	Property	doc_id
1	author	authorid	ID	ex.xml
2	article	authorID	IDREF	ex.xml
3	editor	name	CDATA	ex.xml

<그림 5>ElementInfo테이블과 AttributeInfo테이블의 예

5. 결론

본 논문은 XML 문서의 RDBMS 저장시 XML과 RDBMS의 구조의 불일치에서 오는 데이터의 단편화, 과도한 릴레이션의 생성 등의 문제를 해결하기 위해 DTD 그래프를 바탕으로한 효율적인 릴레이션과 속성으로의 사상방법을 연구하였다. 그리고 XML 문서는 객체 기반의 구조이기 때문에 다중값을 갖는 attribute와 엘리먼트의 순환 등과 같은 문제를 해결하기 위한 별도의 저장방법이 필요하다. 따라서, 이러한 문제를 해결하면서 저장공간의 낭비를 최소화한 저장방법을 제시하였다. 그리고 element들간의 관계 분석을 통한 저장방법을 제시하여 릴레이션으로 사상함으로써, 저장되어진 XML 데이터에 대한 질의의 효율성을 높이고자 하였다. 또한, XML 문서의 복원과 데이터베이스의 접근 횟수를 줄이기 위한 방법으로 DTD와 각 테이블에 대한 정보를 위한 별도의 테이블을 생성하였다.

참고문헌

[1] R. Bourret, "Mapping DTDs to Databases", www.xml.com/pub/a/2001/05/09/dtdtodbs.html, 2001

[2] R. Bourret, C. Bornhovd, A. Buchmann, "A Generic Load/Extract Utility for Data Transfer Between XML Documents and Relational Databases", TR-DVS99-1, DVS, Dep.CS, Darmstadt U. of Technology, G ermany, Dec. 1999

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

[4] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, J. Naughton, "Relational Databases for Querying XML Documents : Limitations and Opportunities", VLDB, Edinburg, Scotland, 1999