XML 문서를 위한 DTD 독립적인 데이터 모델 설계

김정은*, 신판섭*, 이재호**, 임해철*
*홍익대학교 컴퓨터공학과
**인천교육대학교 컴퓨터교육과
{jekim, psshin, lim}@cs.hongik.ac.kr, jhlee@mail.inue.ac.kr

Design of DTD-independent Data Model for XML Document

JeongEun Kim*, PanSeop Shin*, Jaeho Lee**, HaeChull Lim*
*Dept. of Computer Engineering, Hong Ik University
**Dept. of Computer Education, Inchon National University of Education

요 약

XML은 사용자가 문서를 적절히 구조화해서 저장시킬 수 있는 장점으로 인해 기존의 HTML로 구축된 웹서비스 환경을 데이터베이스화하여 다양하고 복잡한 검색 환경을 제공한다. 이러한 이유로 최근에 XML 문서를 데이터베이스에 저장하고 효율적인 관리 및 검색을 지원하는 연구들이 많이 진행되고 있다. 기존 연구를 살펴보면 XML의 구조적 특징을 문서 독립적으로 모델링하지 않아 갱신 및 검색 효율이 떨어지고, 생성되는 테이블의 수가 증가하며, 원문 복원 능력의 저하와 데이터 중복이 발생하는 문제점을 안고 있다. 따라서 본 논문에서는 관계 데이터베이스를 기반으로 XML문서의 구조 정보를 DTD 독립적으로 구성하여 테이블 생성 수와 갱신의 문제점을 개선하고, DTD 구조의 경로 정보 표현을 제안하여 탐색 및 원문 복원 능력을 강화하며, 데이터 중복 문제를 해결한 데이터 모델을 제안한다.

1. 서 론

최근, 인터넷의 발전은 많은 사용자들로부터 기존 정보의 공유와 호환성, 편의성 등을 필요로 하게 되었으며, 이에 힘입어 정보 전달의 중요 수단으로 HTML(HyperText Markup Language)이 부각되어왔다. 그러나 HTML은 쉽게 웹서비스 문서를 작성할 수 있는 장점을 가지고 있는 반면, 고정된 하나의 문서구조만 가질 수 있어 문서 구조 서술의 제약이 많고, 효과적인 문서 검색이 어렵다는 단점을 지니고 있다. 이러한 이유로 1997년 W3C는 HTML의 편리한 사용성과 SGML의 확장성 등의 장점을 취합하여 XML(Extensible Markup Language)을 정의하였다.

XML은 문서의 구조를 정의하는 부분으로 DTD 기술 방법에 대한 표준안을 제공하며, 이밖에도 XML과 인터넷 문서간의 링크를 정의하는 XLL(XML Linking Language), 그리고 실제 XML문서의 외부 표현 형식을 정의하는 XSL(XML Stylesheet Language)로 구성되어 있다. 따라서, 구조적으로 좀더 완벽하고 체계적인 문서 정의가 가능해지고, 하나의 XML 문서에 대해 적용되는 XSL 포맷에 따라 다양한 형태의 문서를 표현할 수 있으며, XLL에 의해 강력하고 다양한 링크 표현을 지원하게 되었다. 이러한 이유로 최근에는 XML 문서를 데이터베이스에 저장하고 효율적인 관리 및 검색을 지원하는 연구들이 많이 진행되고 있다.

기존의 데이터베이스 기반의 XML 문서 저장 시스템 연구는 크게 XML 문서를 관계 또는 객체지향 데이터 모델로 변환하여 저장하는 모델링 연구와 효율적인 검색을 위한 인덱스 및 검색 구조 설계 연구로 나누어진다[1][3]. 특히 데이터 모델 변환 연구는 전체 시스템의 저장 및 검색 효율과 응용 프로그램 지원 측면에 매우 중요한 영향을 미친다.

본 연구는 한국과학재단 특정기초연구과제 (과제번호 : 98-0102-09-01-3)의 지원을 받았음.

기존 연구를 살펴보면 다음과 같은 문제점을 발견할 수 있다. 첫째, 기존 시스템들은 XML 문서를 데이터 모델로 변환하는 과정에서 XML의 구조적 특징과 내용을 문서 독립적으로 모델링하여 저장하지 않으므로 XML 문서 구조의 갱신이 발생할 경우, 이에 따른 모든 구조정보가 수정되어야 하는 문제점을 지닌다[5]. 둘째, 기존 저장 시스템은 DTD 정보를 저장함에 있어 DTD 트리의 순차적인 탐색을 기반으로 리스트 형태의 정보를 데이터베이스에 저장하므로, 검색시의 효율이 떨어지거나 임의 위치의 엘리먼트 탐색이 어려운 단점을 가지고 있다. 셋째, 기존 시스템은 XML 문서를 DTD별로 나누어 테이블을 구성하므로 XML문서를 저장하기 위해 생성되는 테이블의 수가 많아지고[4], 이들을 검색할 때 발생하는 다량의 조인 연산으로 인해 검색 속도가 저하되는 문제점을 지닌다[3]. 넷째, 기존 연구에서는 특정 엘리먼트 검색에 비해 검색된 결과를 바탕으로 원문 전체를 복원하는 능력이 떨어지는 경우가 많아 문서 원문과 엘리먼트의 내용을 모두 저장하므로 데이터 중복이 발생한다[3]. 이러한 몇 가지의 문제점들은 전체적인 저장 구조와 검색 및 갱신 연산의 비효율성을 수반할 뿐만 아니라 데이터베이스 시스템의 장점을 최대로 활용하지 못하는 결과를 초래한다. 따라서 본 논문에서는 관계형 데이터베이스를 기반으로 XML 문서의 구조 정보를 문서와 독립적인 스키마로 구성하여 테이블 생성 수와 갱신의 문제점을 해결하고, DTD의 루트 엘리먼트로부터의 경로 정보를 이용하여 탐색 및 원문 복원 능력을 강화하고 데이터 중복을 해결할 수 있는 데이터 모델을 제시한다.

2. 관련연구

XML 문서를 데이터베이스에 저장하는 기존 연구를 살펴보면, XML 문서를 데이터 모델로 사상하여 저장하는 방법에 따라 크게 '가상 분할'

과 '분할 저장'으로 나눌 수 있다[6].

가상 분할 방법은 문서의 전체 내용을 하나의 CLOB으로 데이터베이스에 저장하고, 문서에 내포된 엘리먼트 태그들의 구조(DTD 구조)를 추가적인 위치정보로 표현하여 저장한다. 즉, XML 문서의 엘리먼트들을 추출하고, 그 엘리먼트가 실제 문서상에 어디에 위치하는지를 시작오프셋과 종료오프셋을 사용하여 표현한다. 또 다른 방법인 분할 저장은 문서의 내용을 DTD를 기준으로 엘리먼트와 속성으로 나누어 저장하고 검색 시, 구조 정보를 참조하여 해당 엘리먼트나 하위 엘리먼트들의 조합을 생성하여 처리한다.

가상 분할은 DTD와 독립적인 데이터 모델을 사용하므로 생성되는 데이터베이스 테이블의 수와 데이터의 중복이 적고 검색 효율이 우수한 특징을 가지고 있다. 검색 구조의 측면에서 볼 때, 하위 엘리먼트의 시작오프셋은 상위 엘리먼트의 시작오프셋보다 크고 종료오프셋은 상위 엘리먼트의 종료오프셋보다 작으므로, 엘리먼트에 대한 검색이 발생할 경우 전문에서 위치 정보만을 사용하여 구조적인 검색이 가능하다는 장점을 가지고 있다. 그러나, XML문서의 엘리먼트가 추가, 삭제되는 등의 갱신이 발생하면, 다른 엘리먼트들의 위치 정보가 모두 수정되어야 하므로 이 때문에 발생하는 오버헤드와 데이터베이스의 일관성 유지가 큰 문제로 대두된다. 따라서 가상 분할 기법은 검색 위주의 응용 개발에 적합한 형태라 하겠다[2].

반면, 분할 저장의 경우, 문서를 DTD의 엘리먼트 별로 나누어 저장하므로, 특정 엘리먼트 갱신 시, 이 엘리먼트와 관련된 부모, 자식, 전, 후 엘리먼트의 정보만 변경해 주면 되므로 가상 분할 방법보다 문서 내용의 갱신이 자유롭다는 장점을 가지고 있다. 그러나, 분할 저장 방법은 DTD에 의존적인 데이터 모델을 사용하므로, DTD 구조 자체의 갱신이 발생하면 이와 관련된 데이터베이스 테이블을 모두 재구성해야 하는 오버헤드를 지니고 있다. 또한 분할 저장 방법을 사용하여 검색을 할 경우, 검색 대상이 되는 엘리먼트가 다른 엘리먼트를 포함하는 상위 엘리먼트라면, 그 하위 엘리먼트를 모두 조합하여 결과를 구성해야 하므로 검색 과정이 복잡하고 검색 시간이 길다는 단점을 가지고 있다[4].

본 논문에서는 XML 문서의 갱신이 비교적 자유로운 분할 저장 방법을 기반으로, DTD 독립적인 관계형 스키마를 정의하고, 리프 노드에서 해당 엘리먼트까지의 경로 정보를 인스턴스로 갖는 DTD 구조 테이블을 제안하여 분할 방법의 검색 비효율성을 해결한다.

3. 데이터 모델 설계

본 장에서는 XML 문서와 DTD 정보를 분리하여 DTD와 독립적인 관계 데이터 모델로 변환하는 사상 기법을 제안한다. 제안된 데이터 모델의 스키마는 모두 6개의 릴레이션으로 구성되어 있으며, DTD와 XML 문서의 모든 정보를 저장하고 관리 가능하도록 설계하였다. 제안된 스키마를 구성하는 릴레이션은 <그림 1>과 같이 구성된다.

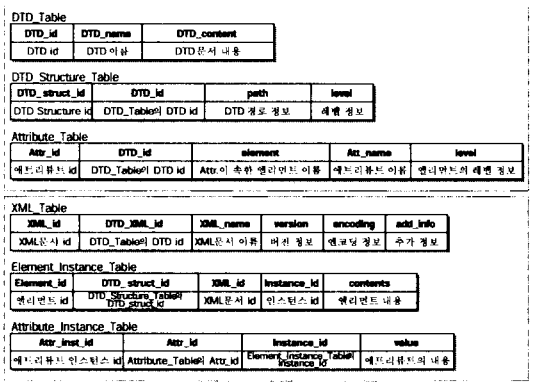

그림 1 관계 저장 스키마 구성

XML 문서를 이루는 모든 DTD는 DTD_Table에 등록되며, 이러한 DTD에서 추출된 구조 정보는 DTD_Structure_Table에 경로 정보와 함께 저장된다. 그리고 이와 관련된 애트리뷰트 정보는 Attribute_Table에 저장된다. XML_Table에는 XML문서의 기본적인 정보가 저장되며, XML 문서의 내용정보는 Element_Instance_Table, Attribute_Instance_Table에 저장된다.

3.1 XML 문서 저장 스키마

DTD와 독립적으로 XML 문서를 저장하고 구조적 검색이 용이하도록 DTD_Table과 DTD_Structure_Table, Attribute_Table을 별도로 구성한다. 이 세 개의 테이블에는 DTD에서 추출할 수 있는 모든 정보가 저장된다. DTD_Table은 저장되는 문서의 모든 DTD에 id를 부여하고, 그 DTD에 따라 작성된 XML 문서를 XML_id를 사용하여 표현한다. DTD_Structure_Table에는 DTD 명세에서 가능한 모든 경로 정보를 엘리먼트별로 구분하여 path필드에 저장한다. 경로에 대한 정보를 엘리먼트 단위로 나누어 저장하므로 XML 문서의 갱신 시, 추가적인 정보(오프셋정보 등)의 수정 없이 해당 엘리먼트와 관련된 부분만 갱신이 이루어지게 되어 문서 내용의 수정이 자유롭다. 또한 어떠한 DTD 정보를 저장하더라도 스키마에 정의된 6개의 테이블만을 사용하므로 DTD의 갱신이 발생하여도 DTD 관련 테이블의 인스턴스 갱신으로 처리된다. 그 밖에도 특정위치에 대한 탐색 요구나 하위 엘리먼트를 포함하고 있는 엘리먼트에 대한 검색 요구가 발생했을 때, DTD_Structure_Table에 경로 정보를 이용하여 직접 해당 엘리먼트로 접근이 가능해 효율적인 검색이 지원된다. Attribute_Table에는 DTD에 표현된 애트리뷰트를 추출하여 애트리뷰트 이름과 포함 엘리먼트를 저장한다. <그림 2>는 본 연구에서 제안한 경로 정보의 표현 방식을 도식화한 것이다.

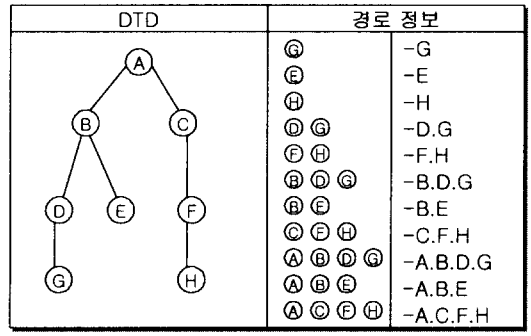

그림 2 DTD에 따른 경로 정보 표현

XML_Table에는 문서 자체에 대한 파일명과 버전정보, XML 문서가 따르는 DTD_XML_id를 저장하고, Element_instance_Table에는 엘리먼트가 속한 XML 문서의 XML_id 필드와 각 엘리먼트의 경로를 나타내기 위해 DTD_struct_id 필드를 사용하여 엘리먼트의 내용을 저장한다. 또한, 반복되는 엘리먼트를 구분하기 위해 동일한 엘리먼트 집합에는 동일한 Instance_id를 부여한다. Attribute_Instance_Table에는 애트리뷰트의 내용 정보를 저장한다.

이와 같이 XML 문서의 내용과 DTD의 구조정보를 구분하여 저장함으로써 DTD마다 생성되는 전체 테이블 수를 줄이고, 검색 시 발생하는 조인 연산을 감소시켰다. 또한 자유로운 갱신 연산이 가능하고 경로 정보를 이용한 원문 복원 능력과 검색 성능을 강화하여 데이터 중복을 해결하였다.

3.2 XML문서 저장 예

<그림3>의 DTD는 서점의 도서검색을 위한 도서정보로서, book, title, author, price와 관련된 7개의 엘리먼트와 하나의 애트리뷰트로 구성되어 있으며, XML문서에는 이 DTD를 따르는 두 권의 도서정보를 담고 있다.

<그림 3>의 DTD와 XML 문서를 본 연구에서 제안한 관계 스키마로 사상하는 과정을 표현하면 다음과 같다.

```
<!ELEMENT bookstore (book)*>
<!ELEMENT book (title,author*,price)>
<!ATTLIST book genre CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (name | (first -name,last-name))>
<!ELEMENT price (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT first -name (#PCDATA)>
<!ELEMENT last -name (#PCDATA)>
```

```
<?xml version='1.0'?>
<bookstore>
 <book genre="autobiography">
  <title>The Autobiography of Benjamin Franklin</title>
  <author>
   <first -name>Benjamin</first -name>
   <last-name>Franklin</last-name>
  </author>
  <price>8.99</price>
 </book>
 <book genre="philosophy">
  <title>The Gorgias</title>
  <author>
   <name>Plato</name>
  </author>
  <price>9.99</price>
 </book>
</bookstore>
```

그림 3 도서정보 DTD와 XML문서 예

DTD_Table에는 DTD 문서의 id와 그 문서의 이름인 "book.dtd", 그리고 DTD 문서의 내용이 저장된다. <그림 4>는 DTD_Table의 저장 정보를 나타낸 것이다.

DTD_id	DTD_name	DTD_content
1	book	...

그림 4 DTD_Table

DTD_Structure_Table에는 "book.dtd"의 리프 엘리먼트인 name, first_name, last_name, title, price로부터 루트 엘리먼트인 bookstore까지의 경로 정보를 상위 엘리먼트에서 하위 엘리먼트 순으로 조합하여 저장한다. <그림 5>는 저장된 경로정보를 나타낸다.

DTD_struct_id	DTD_id	path	level
1	1	name	3
2	1	first_name	3
3	1	last_name	3
4	1	title	2
5	1	author.name	2
6	1	author.first_name	2
7	1	author.last_name	2
8	1	price	2
9	1	book.title	1
...	...	...	...
18	1	bookstore.book.price	0

그림 5 DTD_Structure_Table의 저장된 경로 정보

Attribute_Table에는 애트리뷰트를 가지는 엘리먼트 정보가 저장되므로 <그림 6>의 예를 보면 book과 book의 DTD 구조의 레벨값인 "1"이 Level필드에 저장되고, Attr_name필드에는 엘리먼트 book이 가지고 있는 애트리뷰트의 이름 "genre"가 저장된다.

Attr_id	DTD_id	element	Attr_name	level
1	1	book	genre	1

그림 6 Attribute_Table의 저장 예

XML_id	DTD_XML_id	XML_name	version	encoding	add_info
1	1	book	1.0		

그림 7 XML_Table의 저장 정보

XML_Table에는 XML 문서가 따르는 DTD 정보로 DTD_XML_id필드에 DTD_Table의 DTD_id "1"을 저장하고, XML 문서 이름 "book.xml"을 저장한다. <그림 7>은 XML_Table을 나타낸 것이다.

Element_Instance_Table의 DTD_Struct_id필드는 DTD_Structure_Table에 저장된 경로 정보 리프 엘리먼트에서 해당하는 필드의 id를 저장한다. <그림 8>은 Element_Instance_Table의 저장 예로, Element_id가 "4"인 튜플의 경로정보는 DTD_Structure_Table의 DTD_struct_id "18"번을 참조하여 표현 가능하고, 이 튜플은 XML_Table의 등록된 XML문서 "1"에 포함된 엘리먼트임을 나타내며, 그 내용은 contents 필드의 저장된 "8.99"이다.

Element_id	DTD_struct_id	XML_id	Instance_id	contents
1	4	1	1	"The Autobiography of Benjamin Franklin "
2	2	1	1	"Benjamin "
3	3	1	1	"Franklin "
5	4	1	2	"The Gorgias "
6	1	1	2	"Plato "
7	8	1	2	"9.99"

그림 8 Element_Instance_Table의 저장 정보 예

Attribute_Instance_Table에는 엘리먼트 book의 애트리뷰트 "genre"의 실제 내용인 "autobiography"와 "philosophy"가 저장된다. <그림 9>는 Instance_id가 1이고 book엘리먼트의 "genre" 애트리뷰트 값이 "autobiography"인 튜플과 연관된 엘리먼트의 내용으로 Element_Instance_Table의 Element_id 1, 2, 3, 4를 저장한다.

Attr_inst_id	Attr_id	Instance_id	value
2	1	2	"philosophy"

그림 9 Attribute_Instance_Table의 저장 내용 예

4. 결 론

XML 문서를 저장하는 기존 시스템들은 XML의 구조적 특징과 내용을 문서 독립적으로 저장하지 않으므로 XML 문서 구조의 갱신이 어렵고, DTD 트리의 순차적인 탐색을 기반으로 리스트 형태의 정보를 데이터베이스에 저장하므로, 검색시의 효율이 떨어진다. 또한 기존 시스템은 XML 문서를 DTD별로 나누어 테이블을 구성하므로 XML 문서 저장을 위한 테이블의 수가 증가하여 검색 속도가 저하되는 문제점을 지니고, 원문 전체의 복원을 위해 문서 원문과 엘리먼트의 내용을 모두 저장하므로 데이터 중복이 발생한다. 이러한 몇 가지의 문제점들을 해결하기 위해 본 논문에서는 XML문서의 갱신이 비교적 자유로운 분할 저장 방법을 기반으로, DTD 독립적인 관계형 스키마를 정의하여 테이블 생성 수와 갱신의 문제점을 해결하고, DTD의 리프 노드에서 상위 엘리먼트까지의 경로 정보를 인스턴스로 갖는 DTD 구조 테이블을 제안하여 탐색 및 원문 복원 능력을 강화하였으며, 데이터 중복을 해결할 수 있는 데이터 모델을 제시하여 분할 저장 방법의 검색 비효율성을 해결하였다.

5. 참고 문헌

[1] Takeyuki Shimura, Masatoshi Yoshikawa, Shunsuke Uemura "Storage and Retrieval of XML Documents Using Object-Relational Databases" *DEXA99*, pp. 206-217, 1999

[2] Susan Malaika, "Using XML in Relational Database Applications." *ICDE99*, pp.167, 1999

[3] Jayavel Shanmugasundaram, Kristin Tufte, Chun Zhang, Gang He, David J. DeWitt, Jeffrey F. Naughton, "Relational Databases for Querying XML Documents: Limitations and Opportunities" *VLDB 99*, pp.302-314, 1999

[4] Daniela Florescu, Donald Kossmann, "Storing and Querying XML Data using an RDMBS" *IEEE Data Engineering Bulletin 22(3):* pp.27-34, 1999

[5] H. V. Jagadish, "Review - Relational Databases for Querying XML Documents: Limitations and Opportunities" *ACM SIGMOD Digital Review 1:*, 1999

[6] Francois, P., "Generalized SGML repositories: Requirements and modeling" *Computer Standard & Interface*, 1996