XML 문서 관리를 위한 혼합 저장 구조 설계

황종욱○ 정재희 강현석
경 상 대 학 교 컴 퓨 터 과 학 과
{goodday, skrkwk, hskang}@dblab.gsnu.ac.kr

Design of A Hybrid Storage Structure For Managing XML Documents

Jong-Wook Hwang○ Jae-Hee Jeung Hyun-Syug Kang
Dept. of Computer Science, Gyeongsang National University

요 약

XML로 기술된 전자 문서를 논리적 구조에 따라 분할하여 객체 지향 데이터베이스에 저장하기 위한 연구가 많이 이루어지고 있다. 그러나, 그러한 접근은 몇몇 기본적인 접근 연산에 대해 성능이 떨어진다. 이 경우, 비분할 저장 구조 모델을 이용하면 이러한 문제를 어느 정도 보완할 수 있다.

본 논문에서는 구조화된 XML 문서의 효율적인 관리를 위해 혼합 저장 구조 모델을 제안한다. XML 문서를 분할과 비분할 모델이 혼합된 형태의 물리적 저장 구조로 구조 정보를 표현하면서 투명성을 제공하기 위한 객체 지향 메타 스키마를 제안하고, 이 메타 스키마로부터 동적으로 생성된 응용 데이터베이스 스키마를 통해 구조화된 문서를 객체 지향 데이터베이스에서 관리하는 방법을 제안한다.

1. 서론

XML(eXtensible Markup Language)은 1998년 W3C(World Wide Web Consortium)에 의해서 표준으로 채택된 전자문서의 기술을 위한 메타 언어(Meta Language)이다. XML은 SGML(Standard Generalized Markup Language)의 복잡한 면을 제거하고 구조, 검증, 확장의 특징을 계승하면서 현재 웹 문서 기술 언어로서 널리 사용되고 있는 HTML의 단점을 보완하여 차세대 웹 문서의 표준으로 자리잡고 있다[1,2].

XML은 이 기종 시스템들간의 정보 교환을 용이하게 하는 등 그 특성 때문에 CALS, EC/EDI, 전자도서관, 전자상거래 등 다양한 분야의 응용에서 채택되고 있다. 이에, XML 문서의 효율적인 저장, 관리 및 검색에 관한 다양한 연구가 이루어지고 있다.

XML 문서를 데이터베이스에 저장하기 위한 기존의 연구들을 살펴보면, DTD의 구조 정보를 해석하여 각 엘리먼트 단위의 클래스를 생성시켜 문서의 논리적 단위를 대응되는 클래스의 인스턴스 트리형태의 완전히 구조화된 데이터베이스 내부 표현을 가지는 분할 저장 구조와 문서 자체를 하나의 연속된 스트링으로 취급하여 BLOB 형태로 저장하여 특정 엘리먼트에 대한 접근은 가지거리(offset)를 활용하는 비분할 저장 구조로 크게 분류할 수 있다. 그러나 분할 저장 접근 방법은 문서의 수정이 용이한 반면, 전체 문서의 삽입 및 검색 등 기본적인 연산이 고비용이며 비분할 저장 접근 방법은 저장 비용 및 기본적인 연산에 있어서 향상된 성능을 달성할 수 있겠지만, 갱신 등 완전히 구조화된 표현이 적절한 문제에 있어서는 결점을 지니게 된다. 따라서, 이 두 가지 접근 방법의 극단적인 결점을 회피하기 위해서 분할과 비분할 방법을 혼합한 접근 방법에 의한 XML 문서의 저장에 관한 연구가 요구되어진다.

한편, DTD는 일반적으로 데이터베이스 스키마로 활용이 가능하지만 DTD는 특정 타입의 문서 구조만을 규정하고 있지 개개의 엘리먼트의 물리적 표현과 속성 타입의 의미에 관한 정보를 지니지 못하고 있다. 따라서 적절한 데이터베이스 내부의 혼합 표현을 위해서는 DTD를 확장, 변형해 데이터베이스 스키마로 나타낼 수 있는 메타 스키마가 필요하다. 이러한 메타 스키마에는 엘리먼트의 물리적 표현 및 인덱싱 등에 관한 정보를 포함할 수 있다.

본 논문에서는 이러한 메타 스키마를 제시하고 이 메타 스키마를 바탕으로 하여 동적으로 생성된 응용 스키마를 만들어 XML 문서를 저장하는 방법을 객체 지향 데이터베이스를 기반으로 설계하였다.

2. 관련연구

XML과 같은 구조적(structured) 문서를 저장/관리하기 위한 연구들이 많이 이루어지고 있다. 이러한 연구들을 DBMS 측면에서 살펴본다면, 크게 객체 지향 데이터베이스를 기반으로 하는 객체 지향 모델과 관계형 데이터베이스를 기반으로 하는 관계형 모델로 나눌 수 있다.

객체 지향 모델 접근의 경우, [4]에서는 SGML 문서를 O_2의 객체 지향 스키마로 표현하고 질의 언어로 O_2SQL을 확장시켜 제시하고 있으며 [9]에서는 DTD를 기반으로 한 응용 스키마의 동적 생성과정을 보여주고 있다. [7]에서는 단말 노드를 독립된 객체로 생성시키지 않고 바로 상위의 비단말 노드의 속성으로 표현하여 객체의 수를 줄이고 있다. 이러한 객체 지향 모델은 객체 지향 데이터베이스 시스템의 풍부한 모델링 능력을 활용

하여 문서의 구조를 효율적으로 표현할 수 있고 문서의 수정을 용이하게 하지만, 문서 집합의 성장에 따른 객체 집합의 가파른 증가로 전체 문서에 대한 기본적인 연산에 있어서는 비효율적이다.

관계형 모델 접근은 관계형 데이터베이스의 스키마인 테이블로 모든 데이터를 모델링하므로, 계층 구조를 가지는 트리로 표현될 수 있는 XML 문서를 모델링하기에는 적합하지 않을뿐더러, 문서를 재구성하기 위한 다수의 테이블에 대한 조인 연산 비용이 증가한다. [3]에서 관계형 데이베이스 시스템에 전통적인 텍스트 관리 시스템을 혼합하여 SGML 문서를 하나의 단일 텍스트 단위로 다루며 검색을 위한 세 종류의 가상 테이블을 정의하고 있다. [8]에서는 변형된 OEM 모델로 표현되는 반구조적인 데이터 모델을 관계형 모델로 사상하며, 그러한 사상을 STORED라는 질의 언어로 표현하고 있다. [5]에서 또한 문서내의 구조적 정보의 표현을 위한 스키마로 테이블을 사용하고 있다.

3. 메타 데이터베이스 스키마

다음은 전형적인 XML 문서로서 (그림 1)은 DTD를 (그림 2)는 이 DTD에 맞게 발생된 문서 인스턴스이다.

```
<!DOCTYPE FAQ [
<!ELEMENT FAQ          (INFO, PART+)>
<!ELEMENT INFO         (SUBJECT, AUTHOR,
                        EMAIL?, VERSION?, DATE?)>
<!ELEMENT SUBJECT      (#PCDATA)>
<!ELEMENT AUTHOR       (#PCDATA)>
<!ELEMENT EMAIL        (#PCDATA)>
<!ELEMENT VERSION      (#PCDATA)>
<!ELEMENT DATE         (#PCDATA)>
<!ELEMENT PART         (Q+)>
<!ELEMENT Q            (QTEXT, A)>
<!ELEMENT QTEXT        (#PCDATA)>
<!ELEMENT A            (#PCDATA)>
<!ATTLIST PART NO      CDATA #IMPLIED
                       TITLE CDATA #IMPLIED>
<!ATTLIST Q    NO      CDATA #IMPLIED>
]>
```

(그림 1) 예제 DTD

```
<?xml version= 1.0' encoding="UTF -8"
                        standalone='yes'?>
<!DOCTYPE FAQ SYSTEM 'FAQ.DTD'>
<FAQ>
<INFO>
  <SUBJECT> XML </SUBJECT>
  <AUTHOR> Lars Marius Garshol </AUTHOR>
  <EMAIL> larsga@ifi.uio.no </EMAIL>
  <VERSION> 1.0 </VERSION>
  <DATE> 29 Jun 97 </DATE>
</INFO>
<PART NO='1'>
  <Q NO='1'>
    <QTEXT> What is XML? </QTEXT>
    <A> SGML light. </A>
  </Q>
  <Q NO='2'>
    <QTEXT> What can I use it for? </QTEXT>
    <A> Anything. </A>
  </Q>
</PART>
```

(그림 2) 예제 문서

이러한 XML 문서를 데이터베이스에 혼합된 내부 표현을 가지도록 XML 문서를 저장하기 위해서는 응용 DTD와 의미적 수준에서 대응되면서 문서의 실제적인 물리적 표현을 위한 부가적인 정보가 반영된 형태의 메타 스키마로의 변환이 요구된다. 본 논문에서는 그러한 메타 스키마를 XML로 표현하고 메타 스키마를 위한 소위, 메타 DTD를 정의한다. 다음의 (그림 3)은 이러한 메타 DTD의 일부이고 (그림 4)는 (그림 1)과 같이 기술된 XML 문서의 물리적 저장 방법을 반영한 메타 DTD의 한 인스턴스인 메타 스키마의 일부이다.

```
<!ELEMENT elementtype ((any|empty|model),( attdef)*)>
<!ATTLIST elementtype name NMTOKEN #REQUIRED
                      FLAT (yes|no) "no" ... >

<!ELEMENT model ( pcdata|element|mixed|choice|sequence|  parameter )>

<!ELEMENT attdef (enumeration?,(required|implied|default|fixed))>
<!ATTLIST attdef name NMTOKEN #REQUIRED
                 datatype NMTOKEN #REQUIRED     >
```

(그림 3) 메타 DTD의 일부

문서의 구성요소들이 어떠한 물리적 표현을 갖게 될지는 문서 구성요소의 의미를 가장 잘 아는 DTD 설계자에 의해 결정될 것이다. (그림 4)에서 보듯이 모든 엘리먼트는 'FLAT'이라는 속성을 가지며, 그 값이 'yes'이면 파서에 의해 해석되어 해당 엘리먼트를 루트로 하여 트리 구조상의 하위 엘리먼트에 대응되는 객체를 각각 생성하는 대신에 하위 엘리먼트는 하나의 데이터베이스 객체내의 속성값으로 표현된다. 그 외에 인덱스를 위한 속성도 정의된다.

```
<elementtype  name= "FAQ">
  <model>
    <sequence>
      <element name=  "INFO">
      <element name=  "PART", occurs= "0,*">
    </sequence>
  </model>
  <attdef  name= "FLAT" datatype ="NMTOKEN ">
    <enumeration><option>yes</option><option>no</option></enu    meration>
    <default>no</default>
  </attdef >
</elementtype >
<elementtype  name= "INFO">
  <model>
    <sequence>
      <element name=  "SUBJECT ">
      <element name=  "AUTHOR ">
      <element name=  "EMAIL " occurs= "0,1">

</elementtype >
<elementtype  name= "PART ">
  <model><element name=  "Q" occurs= "1,*"></model>
  <attdef  name= "NO" datatype ="CDATA"><IMPLIED/></ attdef >
</elementtype >
<elementtype  name= "SUBJECT "><model>< pcdata /></model></ elementtype >
```

(그림 4) 메타 스키마의 일부

4. 응용 데이터베이스 스키마의 생성

XML 문서의 저장은 특정 응용 DTD에 독립적인 기반 클래스 계층과 응용 DTD에 의존적인 클래스 계층이 상호 연계된 구조를 갖는다. 문서의 저장시 메타 스키마에 내재된 물리적 표현에 관련된 정보 및 구조 정보가 파서에 의해서 분석되어 각 엘리먼트는 미리 정의된 기반 클래스로부터 확장된 엘리먼트 타입 클래스의 인스턴스화된 객체로 저장된다.

4.1 DTD 독립 기반(base) 클래스 계층

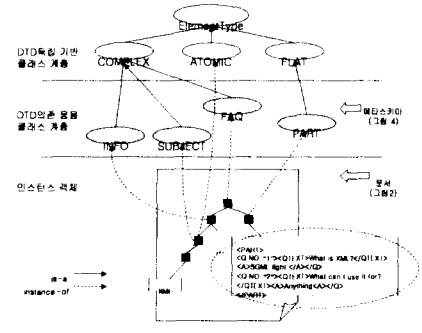
모든 문서 인스턴스에 동일한 구조를 갖는 클래스로 구조 정보의 유지와 관련된 항해 연산이나 속성을 갖는다. DTD에 의존한 클래스를 생성키 위한 슈퍼 클래스이다. 비단말 노드를 위한 'complex', 단말 노드를 위한 'atomic', DTD 설계자에 의해 지정된 비분할 객체를 위한 'flat'클래스가 있다.

4.2 DTD 의존 응용(appliation) 클래스 계층

특정 문서의 응용 DTD와 의미적 수준에서 대응되는 메타 스키마를 해석하여 실제 문서의 저장을 위해 기반 클래스 계층으로부터 확장되어 생성된 클래스를 말한다. DTD마다 다른 구조를 갖는다.

4.3 인스턴스 객체

실제 문서가 동적으로 생성된 DTD 의존 응용 클래스의 인스턴스화된 객체 트리로 표현된다. 그리고 임의 타입의 문서 전체에 대한 참조로써 문서의 루트 엘리먼트에 대응되는 인스턴스 객체의 객체 식별자가 유지된다.



(그림 5) XML 저장의 개괄적 구조

5. 질의 처리

XML 문서에 대한 질의 처리를 위해서는 분할 구조 자체가 가지는 구조 정보만으로는 내용기반 질의나 속성에 관한 질의의 처리는 어렵다. 따라서, 질의의 효과적인 처리를 위해서 역화일로 구성된 내용 인덱스, 속성 인덱스, 엘리먼트 인덱스를 활용하겠지만 여기서는 혼합된 데이터베이스 내부 표현에 대한 엘리먼트 인덱스를 활용한 구조기반 질의 처리에 대해서만 살펴보겠다.

5.1 분할 객체에 대한 질의

각 분할 객체는 부모, 자식, 형제의 객체 식별자를 속성값으로 하는 속성을 가지고 있으므로 참조를 추적하면 쉽게 찾을 수 있다. 따라서 여기서는 상세한 설명은 생략한다.

5.2 비분할 객체에 대한 질의

비분할 객체를 포함하는 질의의 경우 객체 식별자만으로 비분할 객체내의 논리적 구조에 대한 질의가 어려움으로 다음과 같은 구조의 엘리먼트 인덱스로 비분할 객체내의 엘리먼트의 논리적 구조를 표현한다.

Oid	Element	Offset	Length	Parent
3	PART	1	124	0
	Q	7	51	1
	QTEXT	10	26	7
	A	37	17	7
	Q	59	58	1
	QTEXT	62	35	59
	A	98	15	59

(그림 6) 비분할 객체내의 논리적 구조의 표현

(그림 5)의 점선으로 둘러진 비분할 객체내의 특정 엘리먼트에 대한 식별은 (그림 6)과 같이 비분할로 표현된 객체의 객체 식별자와 루트 엘리먼트에서 대상 엘리먼트까지의 가지거리(offset)에 1을 더한 값으로 표현한다. 'Parent'필드는 트리 구조상의 패스정보를 유지하기 위해 부모 엘리먼트의 가지거리를 값으로 가지는데 그 값이 '0'이란 의미는 비분할 객체내의 부분 트리 구조상의 루트 엘리먼트라는 것을 뜻한다. 테이블의 각 튜플은 가지거리에 의해서 정렬시켜서 특정 패스의 추적시 이진탐색 기법을 적용한다. 예를 들어, (그림 5)에서 점선으로 둘러진 부분이 비분할 객체로 저장되었다고 가정했을 때, 'PART 엘리먼트의 두 번째 자식 엘리먼트의 모든 자식 엘리먼트를 찾아라'는 질의가 주어졌다면, 테이블을 조사하여 PART 엘리먼트의 가지거리를 구한 뒤, 그 가지거리 값을 Parent의 값으로 가지는 두 개의 'Q' 엘리먼트를 찾는다. 동일한 엘리먼트 이름을 갖지만 DFS(Depth First Search)에 의해 테이블의 구성시 상위의 튜플이 하위의 튜플보다 왼쪽에 위치한다는 순서정보를 지니고 있으므로 테이블의 순서상 두번째의 'Q'엘리먼트의 가지거리를 구해서 그것을 Parent의 값으로 갖는 엘리먼트를 추출하면 된다.

6. 결론 및 향후 연구과제

본 논문에서는 문서의 물리적 표현과 관련된 정보의 저장을 위한 메타 스키마를 설계하고 분할과 비분할 저장 접근 방법의 장점을 상호 보완적으로 혼합한 데이터베이스 내부 표현을 위한 XML 저장 스키마를 설계하였다. XML 문서의 혼합된 데이터베이스 내부 표현으로 분할 저장 접근 방법에서의 전체 문서의 삽입과 검색, 저장 비용을 줄이면서 비분할 저장 모델의 수정 비용을 줄일 것이다.

향후 본 설계를 바탕으로 한 저장과 검색 시스템을 구현하고 성능을 평가하여 XML 문서를 객체 지향 데이터베이스를 활용하여 효과적으로 저장하고 검색하는 방법에 대한 추가적인 연구가 필요하다. 그리고 데이터베이스내의 물리적 표현을 위한 정보를 본 논문에서는 DTD 설계자가 규정토록 했으나, 그러한 정보의 미규정시 효율적인 저장 공간의 활용을 위해 문서내 엘리먼트의 역할(role) 및 의미(semantic)를 분석하여 최적화된 저장 스키마의 동적 생성에 관한 연구가 요구되어진다.

7. 참고문헌

[1] W3C, "Extensible Markup Language 1.0," http://www.w3.org/TR/1998/REC-XML-19980210, 1998.

[2] R. Davis, T. Moore, and J. Zobel, "Database Systems for Structured Document," Int'l. Symp. Advanced Database Tech. and their Integration(ADTI '94), pp. 272-283, Nara Japan, 1994.

[3] G. Blake, M. Consens, P. Kilpeläinen, P. Larson, T. Snider, and F. Tompa, "Text/Relational Database Management Systems: Harmonizing SQL and SGML," In Proc. First Int'l, Conf. Appl. of Database, pp. 267-280, June 1994.

[4] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl, "From Structured Documents to Novel Query Facilities," In Proc. ACM SIGMOD Int'l. Conf. Management of Data, pp. 313-324, May 1994.

[5] I. A. Macleod, "Storage and Retrieval of Structured Documents," J. of Information Processing & Management, Vol. 26, No. 2, pp. 197-208, 1990.

[6] K. Aberer, K. Böhm, and C. Hüser, "The Prospects of Publishing and Object Paradigms," Electronic Publishing, Vol. 8(2&3), pp. 63-79, Dec. 1993.

[7] Yangjun Chen, Karl Aberer, "Layered Index Structures in Doument Database Systems," In Proc. ACM 7th Int'l Conf. Information and Knowledge Management, pp. 406-413, Nov. 1998.

[8] Alin Deutsch, Mary Fernandez, Dan Suciu, "Storing Semistructured Data with STORED," SIGMOD '99 Philadelphis PA, pp. 431-442, 1999.

[9] 한에노, 박인호, 강현석, 김완석, "SGML 문서의 관리를 위한 객체지향 데이터베이스 설계," 한국 정보 처리학회 논문지, 4권 3호, pp. 670-684, 1997. 3.