濟州大學校 實驗技術硏究所 論文集, 第14卷, 1號, pp.109-114, 2003. 6
Cheju Nat'l Univ. Res. Inst. Adv. Tech. Jour., Vol.14, No.1, pp.109-114, 2003. 6

# RDBMS를 이용한 XML 문서 저장 시스템 설계

박 충 희* · 도 양 회** · 이 상 준***

# Design of the XML Document Storage System Using RDBMS

Chung-Hee Park* · Yang-Hoi Doh** · Sang-Joon Lee***

## ABSTRACT

In this paper, we propose the data storage model which supports the efficient updates and retrieval for XML documents. For proposed system, we adopt the DTD-independent schema form, and change the virtual fragmentation model for storing XML documents. And we design new identifier(called position ID) which does not update the position values of the other elements after many element insertion and deletion.

**Key Words** : XML, XML document, RDBMS, storage system

## Ⅰ. 서 론

인터넷의 발전에 따라 인터넷상의 정보를 보다 효과적으로 사용하고자 하는 연구가 활발히 진행되고 있으며 XML[6]은 언어 자체가 지니는 정보 표현의 유연성 때문에 인터넷상의 정보 표현 및 교환의 표준으로 각광을 받고 있다. 따라서 나날이 급증하고 있는 XML 문서의 저장과 검색 같은 새로운 형태의 데이터 관리 기술에 대한 요구가 급증하고 있다. XML 문서의 효과적인 저장 및 검색 시스템은 XML 문서의 특성을 활용하기 위하여 구조검색, 내용검색, 속성검색이 이루어져야 하며, 일반 텍스트 문서와 달리 XML 문서를 구성하는 단위인 노드(엘리먼트)의 의미별로 검색 및 갱신이 지원되어야 한다[5]. 한편 XML 문서의 저장소(Repository)로는 다양한 형태의 시스템이 가능하다. 첫 번째로는 관계형 데이터베이스 관리 시스템이나 객체 지향 데이터베이스 관리 시스템과 같이 기존의 데이터베이스 기반의 XML 문서 저장 시스템이 있다[7].

두 번째로는, XML 문서의 저장을 위한 전용 시스템을 사용하는 방법이 있으며[8] 마지막으로 텍스트 파일 시스템을 사용하는 방식이 있다. 이 중에서 기존의 데이터베이스 관리 시스템을 이용하는 방식은 데이터베이스에서 제공하는 여러 기능(질의 최적화, 동시성 제어, 회복 등)을 사용할 수 있고 XML 문서와 데이터베이스에 이미 존재하는 일반적인 데이터를 동시에 이용하는 응용프로그램을 작성하는 데 용이하다는 장점을 가지고 있다. 또한 텍스트 파일 시스템보다 효율적인 XML 문서의 갱신이 가능하다. 반면 분할 저장 모델을 취하는 데이터베이스 이용방식은

* 제주대학교 컴퓨터공학과 대학원
Department of Computer Eng., Cheju Nat'l Univ.
** 제주대학교 전자공학과
Department of Electronic Eng., Cheju Nat'l Univ.
*** 제주대학교 통신컴퓨터공학부, 첨단기술연구소
Faculty of Communication & Computer Eng., Cheju Nat'l Univ., Res. Inst. of Adv. Tech.

XML 문서 검색 시 해당노드들에 대한 정보를 취합하는 과정으로 인하여 검색시간이 상대적으로 길어진다는 것이다. 특히 전체문서를 검색할 경우에는 가상분할 모델 방식(텍스트 파일 시스템)보다 검색시간이 길다는 단점이 있다. 본 논문에서는 관계형 데이터베이스 관리 시스템을 이용하면서도 가상 분할 저장 방식의 장점인 검색시간을 빠르게 지원할 수 있는 XML 문서 저장 방법과 저장된 XML 문서의 검색 처리 과정을 설명한다.

본 시스템에서는 분할 모델에서의 단점인 노드 취합 과정을 줄이기 위하여 단일 테이블 저장 방식을 사용하였고 엘리먼트 추가 또는 삭제시 문서의 구조 정보 변경으로 인한 타 노드의 위치정보 변경 발생을 방지하기 위하여 position id를 제안, 설계하였다 또한 position id를 이용하여 엘리먼트들의 순서 정보를 표현하였으며, 마지막으로 빠른 검색을 위해 기 연구된 구조색인, 내용색인, 속성색인을 변형 지원하였다.

## II. 관련연구

기존의 XML 문서 저장 방법은 XML 문서를 하나의 저장 공간에 저장한 후 특정 노드에 대한 검색이 발생할 경우 노드가 가리키고 있는 저장 공간의 위치(offset)로써 문서의 내용을 검색하는 가상분할 모델 방식과 문서를 구성하는 노드를 노드별로 쪼개어 여러 테이블에 저장하는 분할 모델 방식이 있다[1]. 가상분할 모델 방식인 경우 검색 시 시작위치와 길이 등의 정보로 저장 공간에 대한 단 한번의 접근으로 검색 대상을 가져올 수 있으나 문서의 변경 즉 문서를 구성하는 노드의 추가나 삭제가 발생할 경우 문서를 구성하는 대부분의 노드에 대한 위치정보가 변경되어 많은 양의 노드들에 대해 위치정보를 변경시켜 주어야한다. 이에 비해 분할 모델 방식은 문서 변경이 발생할 경우 다른 노드들에 대한 위치정보를 바꿀 필요 없이 직접 관련된 노드 즉 추가 혹은 삭제 노드만을 변경하지만 전체 문서나 혹은 XML 문서의 일부분을 검색할 경우 여러 테이블에 나뉘어 저장된 노드들에 대한 정보를 취합하여 하나의 문서로 만드는 과정이 필요하게 되어 검색시간은 가상분할 모델 방식에 비해 길어진다[1]. 또한 XML 문서를 관계형 데이터베이스에 저장할 때 발생하는 순서정보의 손실을 방지하기 위하여 특정 노드의 자식노드 순서정보를 테이블 내에 직접적으로 유지하는 방식은 빠른 검색을 지원하지만 노드의 추가나 삭제시 많은 노드들에 대한 순서 정보를 변경해 주어야 한다는 단점이 있다[3].

## III. XML 문서 저장 시스템의 설계

### 3.1. XML 문서 저장 방법

본 연구에서는 DTD의 존재 유무에 관계없이 일반적인 XML 문서들을 저장하기 위하여 XML 문서를 응용 영역별로 나누어 generic 스키마를 생성, 저장한다. 분할 모델방식에서의 단점인 노드 정보 취합 과정을 줄이기 위하여 XML 문서들은 하나의 단일 테이블에 마크업별로 나누어 저장한다. 이때 테이블 내의 노드들의 실제 물리적 저장 순서를 문서내에서 마크업이 나타나는 논리적 순서와 일치시키며 또한 엘리먼트 추가, 삭제시 기존의 타 엘리먼트의 offset 변경을 유발하지 않는 새로운 튜플 id, 일명 position id를 설계하였다.

```
<books>
<book style="textbook">
<title> XML 활용</title>
<editor>
 <fname>민수</fname> <lname>박</lname>
</editor>
<author>
 <fname>현민</fname> <lname>김</lname>
 <fname>상철</fname> <lname>이</lname>
</author>
<summary>
 이 책은 <keyword>XML</keyword> 활용에
관한 교재입니다.
</summary>
</book>
</books>
```

Fig. 1. An example of an XML instance

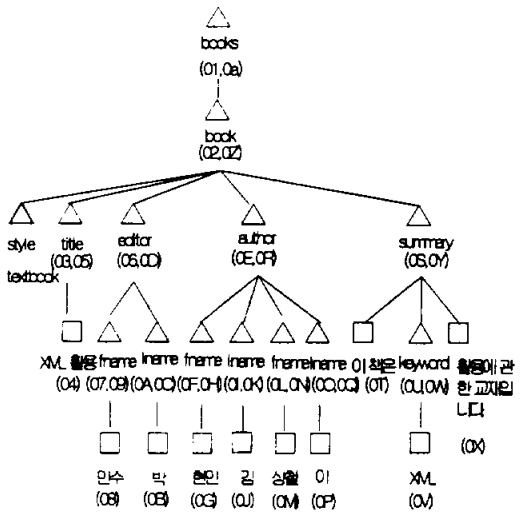


Fig. 2. The tree representation with the position id

Fig. 2는 예제 XML 문서(Fig .1)의 트리 구조 표현과 트리 내 각 노드에 대해 position id가 부여된 형태를 보여주고 있다. position id 번호 부여 방법은 속성노드는 제외하고 루트노드로부터 시작하여 깊이 우선 방식으로 엘리먼트 노드는 시작 태그와 종단 태그에 대해 각각 부여하며, 단말노드는 하나의 값만 부여한다. 이 position id가 기본키(B+-tree)가 되어 문서 내 노드의 논리적 순서와 실제 저장된 순서를 항상 일치시키며 향후 XML 조각이 검색될 경우 빠른 검색이 가능하다.

### 3.2. Position id

본 연구에서 제안하는 position id는 문서 테이블(DT)내 특정노드(엘리먼트)에 해당하는 튜플을 유일하게 식별하는 id 역할을 수행하며 다음과 같은 특성을 갖는다.

① 노드(엘리먼트, 단말)의 삽입, 삭제 후 혹은 단말노드의 문자열 길이 변경과 관계없이 문서내 노드의 논리적 순서와 테이블내 해당 튜플의 저장된 물리적 순서가 항상 같아지는 특성을 갖고 있다.

② 노드 연산 후 타 노드의 position id를 변경할 필요 없음

position id 구성은 두 부분 즉(sid,iid)로 이루어지는 데 여기서 sid는 XML 문서를 파싱하여 문서 테이블에 저장할 때 노드들의 논리적 순서에 따라 초기에 부여되는 일련번호이다. 본 논문에서는 각 문자가 0~9, A~Z, a~z의 값을 갖는 62진법을 사용한다. sid는 5개의 문자열로 구성하였고 iid는 향후 삽입을 위한 부분으로 가변길이의 문자열이다. 연속적으로 position id(pid)가 부여된 노드사이에 XML 문서 조각의 삽입이 발생할 경우 해당 노드들에 대해 문서내의 논리적 순서와 일치되도록 연속적으로 번호를 부여한다. 번호 부여 규칙은 다음과 같다.

① iid는 62진법을 사용하여 1부터 순차적으로 부여되며 0으로 끝날 수 없다. 끝이 0으로 끝나면 1을 더 증가 시킨다.

② f를 삽입 노드의 선행 pid, b를 후행 pid라 할 때, 삽입노드의 pid는 아래와 같이 계산된다.

```
if(value(b) - value(f) = 1)
then if(length(f) < length(b))
        then f & zerostring(b,f) & '1'
        else f & '1'
else if(length(f) < length(b))
        then f & zerostring(b,f) & '1'
        else value(f) +1
```

value(b) - value(f) 연산시 b와 f의 size가 틀리면 0으로 채워 계산하며, zerostring(b,f)는 b의 문자열중에서 f 문자열이후의 연속된 0이나 1의 개수에 해당하는 0을 반환하는 함수이다.

Table 1. The example of numbering using the position id

| 초기 | time1 | time2 | time3 | 논리적 순서 | 비고 |
|---|---|---|---|---|---|
| 00001 | | | | 1 | |
| | | | 0000101 | 2 | '00001'&'0'&'1' |
| | 000011 | | | 3 | '00001'&'1' |
| | | 000012 | | 4 | value('000011')+1 |
| 00002 | | | | 5 | |

### 3.3. 관계 데이터베이스 스키마

제안된 시스템의 스키마는 XML 문서에 대해 DTD 독립적인 generic 스키마를 사용하여 모두 7개의 테이블로 구성하였으며 Oracle 9i를 기준으로 설계하였다. 각각의 문서에 대한 일반사항을 저장하는 영역 테이블(ST), 실제 문서를 저장하는 문서 테이블(DT), 엘리먼트 정보를 저장하는 엘리먼트 목록 테이블(LT), 실제 엘리먼트의 위치정보를 저장하는 엘리먼트 테이블(ET), 그리고 속성을 저장하는 속성 테이블(AT), 내용 정보 검색을 위해 내용을 저장하는 내용 테이블(CT), 노드의 경로를 저장하는 경로 테이블(PT)로 구성된다. 특정 응용에 관련된 모든 XML 문서는 이 7개의 테이블에 저장된다.

| docid | docname | description |
|---|---|---|
| number(4) | varchar2(200) | varchar2(400) |

(a) ST(collection)

| docid | pid | type |
|---|---|---|
| number(4) | varchar2(305) | number(1) |

| content |
|---|
| varchar2(4000) |

(b) DT(document)

| docid | element | etid |
|---|---|---|
| number(4) | varchar2(50) | char(3) |

(c) LT(element list)

| docid | pathexp | pathid |
|---|---|---|
| number(4) | varchar2(300) | number(4) |

(d) PT(path)

| docid | pathid | pid1 |
|---|---|---|
| number(4) | number(4) | varchar2(305) |

| pid2 | ppid1 |
|---|---|
| varchar2(305) | varchar2(305) |

(e) ET(element)

| docid | pathid | attvalue |
|---|---|---|
| number(4) | number(4) | varchar2(200) |

| pid1 | pid2 |
|---|---|
| varchar2(305) | varchar2(305) |

(f) AT(attribute)

| docid | pathid | content |
|---|---|---|
| number(4) | number(4) | varchar2(4000) |

| pid |
|---|
| varchar2(305) |

(g) CT(content)

Fig. 3. the proposed database schema

여기서 엘리먼트 테이블(ET)의 ppid1 속성 값은 특정 엘리먼트의 부모노드의 시작 태그의 position id 값을 갖는다. 특정 엘리먼트의 자식노드의 순서정보 얻기 위하여 명시적으로 저장된 테이블을 사용하지 않고 위의 ppid1 속성을 이용하여 순서 정보를 구한다. 즉 질의 처리시 동일 부모의 자식 노드들(ppid1 값이 동일한 엘리먼트들)을 단지 count 함으로써 순서정보를 구할 수 있다. 이 방식은 엘리먼트 추가, 삭제로 인한 순서정보의 과다한 변경을 방지한다.

### 3.4. XML 문서의 스키마 저장 예

제안한 스키마를 이용하여 Fig. 1의 문서를 저장하였을 때 실제 저장된 테이블의 데이터는 다음과 같다.

| docid | docname | description |
|---|---|---|
| 1 | books.xml | |

(a) ST

| docid | pid | type | content |
|---|---|---|---|
| 1 | 00001 | 1 | <books> |
| 1 | 00002 | 2 | <book style="textbook"> |
| 1 | 00003 | 1 | <title> |
| 1 | 00004 | 9 | XML 활용 |
| 1 | 00005 | 3 | </title> |
| 1 | 00006 | 1 | <editor> |
| 1 | 00007 | 1 | <fname> |
| 1 | 00008 | 9 | 민수 |
| 1 | 00009 | 3 | </fname> |
| 1 | 0000A | 1 | <lname> |
| 1 | 0000B | 9 | 박 |
| 1 | 0000C | 3 | </lname> |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 1 | 0000a | 3 | </books> |

(b) DT

| docid | element | etid |
|---|---|---|
| 1 | author | 001 |
| 1 | book | 002 |
| 1 | books | 003 |
| 1 | editor | 004 |
| 1 | fname | 005 |
| 1 | keyword | 006 |
| 1 | lname | 007 |
| 1 | style | 008 |
| 1 | summary | 009 |
| 1 | title | 00A |

(c) LT

| docid | pathexp | pathid |
|---|---|---|
| 1 | /003 | 1 |
| 1 | /003/002 | 2 |
| 1 | /003/002@008 | 3 |
| 1 | /003/002/00A | 4 |
| 1 | /003/002/004 | 5 |
| 1 | /003/002/004/005 | 6 |
| 1 | /003/002/004/007 | 7 |
| 1 | /003/002/001 | 8 |
| 1 | /003/002/001/005 | 9 |
| 1 | /003/002/001/007 | 10 |
| 1 | /003/002/009 | 11 |
| 1 | /003/002/009/006 | 12 |

(d) PT

| docid | pathid | pid1 | pid2 | ppid1 |
|---|---|---|---|---|
| 1 | 1 | 00001 | 0000a | |
| 1 | 2 | 00002 | 0000Z | 00001 |
| 1 | 4 | 00003 | 00005 | 00002 |
| 1 | 5 | 00006 | 0000D | 00002 |

| docid | pathid | pid1 | pid2 | ppid1 |
|---|---|---|---|---|
| 1 | 6 | 00007 | 00009 | 00006 |
| 1 | 7 | 0000A | 0000C | 00006 |
| 1 | 8 | 0000E | 0000R | 00002 |
| 1 | 9 | 0000F | 0000H | 0000E |
| 1 | 10 | 0000I | 0000K | 0000E |
| 1 | 9 | 0000L | 0000N | 0000E |
| 1 | 10 | 0000O | 0000Q | 0000E |
| 1 | 11 | 0000S | 0000Y | 00002 |
| 1 | 12 | 0000U | 0000W | 0000S |

(e) ET

| docid | pathid | attvalue | pid1 | pid2 |
|---|---|---|---|---|
| 1 | 3 | textbook | 00002 | 0000Z |

(f) AT

| docid | pathid | content | pid |
|---|---|---|---|
| 1 | 4 | XML 활용 | 00004 |
| 1 | 6 | 민수 | 00008 |
| 1 | 7 | 박 | 0000B |
| 1 | 9 | 현민 | 0000G |
| 1 | 10 | 김 | 0000J |
| 1 | 9 | 상철 | 0000M |
| 1 | 10 | 이 | 0000P |
| 1 | 11 | 이 책은 | 0000T |
| 1 | 12 | XML | 0000V |
| 1 | 11 | 활용에 관한 교재입니다. | 0000X |

(g) CT

Fig. 4. A storage example of the XML document

### 3.5. 질의 처리

다음은 주어진 XQL 질의에 대해 변환된 SQL 질의를 보여주고 있다.

질의1: /books//author

```
select e1.pid1, e1.pid2
from  ET e1, PT p1
where e1.pathid = p1.pathid
and  e1.docid = p1.docid
and p1.docid = 1
and  p1.pathexp LIKE '/003%/001'
order by e1.pid1;
```

질의2: //book/author/fname[2]

```
select e2.pid1, e2.pid2
from ( select e1.pid1, e1.pid2, e1.ppid1,
     rank() over (partition by ppid1
     order by pid1 asc) as no
     from ET e1, PT p1
     where e1.pathid = p1.pathid
        and e1.docid = p1.docid
        and p1.docid = 1
        and pathexp LIKE
            '%/002/001/005') e2
where no = 2
order by e2.pid1;
```

동일부모 밑의 자식 노드 중 n번 자식과 동일 엘리먼트명의 형제중 m번 자식은 부모노드의 위치값 즉 ppid1을 기준으로 count하여 계산, 처리한다. 특정 노드의 삭제는 타 노드의 position id에 영향을 미치지 않기 때문에 갱신사항은 발생하지 않고 다만 삭제된 엘리먼트 노드를 참조하는 지식 노드의 부모 엘리먼트 노드 위치 ppid1 값만 조정해 주면된다.

## IV. 결 론

본 연구에서는 관계형 데이터베이스를 이용함에 있어서 기존의 분할 저장 모델방식의 단점을 개선하기 위하여 XML 문서를 마크업 단위로 나누어 단일 테이블에 저장하여 빠른 검색을 지원하였으며 동시에 가상 분할 방식의 단점의 갱신의 비효율성을 개선하기 위하여 노드(엘리먼트드)의 삽입, 삭제후 타 노드의 위치정보(offset) 변경이 필요없는 position id를 제안, 설계하였다. 또한 이를 통하여 XML 문서내 엘리먼트들의 순서 정보를 DB내에 표현, 처리 가능함을 보였다. 향후 연구 과제로는 특정 위치에 과다한 반복적인 엘리먼트 삽입후 길어진 position id의 효과적인 재조정에 관한 연구가 필요하다.

## 참고문헌


1) 연제원, 이강찬, 이규철, 나중찬, 이미영, 1999, 효율적 XML 문서 변경 및 검색을 위한 페이징 기법, 정보과학회 학술발표논문집(I), 제26권 2호, pp.99-101.

2) 연제원, 조정수, 이강찬, 이규철, 1999, XML 문서 구조검색을 위한 저장 시스템 설계, 정보과학회 학술발표논문집(B), 제26권 1호, pp. 3-5.

3) 박종관, 강형일, 손충범, 유재수, 2000, XML 문서에 대한 효율적인 구조 기반 검색을 위한 색인 모델, 정보과학회 학술발표 논문집, 제27권 2호, pp. 18-20.

4) Takeyuki Shimura, Masatoshi Yoshikawa, Shunsuke Uemura, 1999, Storage and Retrieval of XML Documents Using Objected-Relational Databases, DEXA99, pp. 206-217.

5) Tuong Dao, 1998, An Indexing Model for Structured Documents to Support Queries on Content, Structure, and Attributes, Proceedings of ADL '98, pp. 88-97.

6) Extensible markup language(xml) 1.0(second edition), http://www.w3.org/TR/ REC-xml.

7) J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. J. DeWitt, and J. F. Naughtton, 1999, Relational Databases for Querying XML Documents: Limitations and Opportunities, Proceedings of VLDB Conference.

8) Tamino XML server, http://www.software.com/tamino/

9) XML path language (XPath) version 1.0,http://www.w3. org/ TR/xpath