효율적 문서 검색 및 변경을 위한 XML 문서 저장 시스템 설계

박충희* 이상준

제주대학교 통신컴퓨터공학부

chpark_cs@jeju.ac.kr

sjlee@cheju.ac.kr

Design of an XML Document Storage System for Efficient Document Retrieval and Updates

Chung-Hee Park* Sang-Joon Lee

Dept. of Communication & Computer Engineering, Cheju National University

요 약

본 논문에서는 관계형 데이터베이스를 이용하여 XML 문서를 효과적으로 검색 및 갱신을 수행할 수 있는 데이터 저장 모델을 제안한다. 저장 시스템의 스키마는 DTD 독립적인 형태를 채택하였고, 가상 분할 모델과 분할 모델의 장점을 취합한 혼합형태로 설계하였다. 본 시스템에서는 엘리먼트 추가 또는 삭제시 문서의 구조정보 변경으로 인한 변경사항 발생을 방지하기 위하여 타 노드의 위치정보와 독립적인 position id를 사용하였다.

1. 서 론

최근 인터넷의 발전이 진전되면서 인터넷상의 정보를 보다 효과적으로 사용하고자 하는 연구가 활발히 진행되고 있다. HTML의 단점을 극복하고 SGML의 장점을 취한 XML 표준이 등장하면서 특히 XML 문서의 효과적인 저장 및 검색 시스템에 대한 요구가 급증하고 있다. 이러한 저장 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]. 기존의 데이터베이스 기반의 XML 문서 저장 시스템 연구는 XML 문서를 관계 또는 객체지향 데이터 모델로 변환하여 저장하는 모델링 연구[1,2]와 효율적인 검색을 위한 인덱스 및 검색구조 설계연구로 나뉘어진다[3,4,5]. 기존의 모델링 연구에서의 문제점은 XML 문서 구조 갱신시 이에 따른 모든 구조 정보가 수정되어야 하거나 문서 검색시 해당노드들에 대한 정보를 취합하는 과정으로 인하여 검색시간이 상대적으로 길어진다는 것이다[1]. 본 연구에서는 관계형 데이터베이스를 이용하여 XML 문서를 효과적으로 검색 및 갱신을 수행할 수 있도록 혼합된 방식을 사용하였다. 본 시스템에서는 position id를 제안하여 엘리먼트 추가 또는 삭제시 문서의 구조정보 변경으로 인한 타 노드의 위치정보 변경 발생을 방지하였고 또한 분할 모델에서의 단점인 노드 취합 과정을 최대로 줄이기 위하여 단일 테이블 저장 방식을 사용하였다, 그리고 빠른 검색을 위해 구조색인, 내용색인, 속성색인을 지원하였다.

2. 관련연구

기존의 XML 문서 저장 방법은 XML 문서를 하나의 저장공간에 저장한 후 특정 노드에 대한 검색이 발생할 경우 노드가 가리키고 있는 저장공간의 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 검색시 시작위치와 길이 등의 정보로 저장공간에 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]. 한편 특정 노드의 자식노드 순서정보를 테이블 내에 직접적으로 유지하는 방식은 빠른 검색을 지원하지만 노드의 추가나 삭제시 많은 노드들에 대한 순서 정보를 변경해 주어야 한다[3].

3. XML 문서 저장 시스템의 설계

3.1 XML 문서 저장 방법

본 연구에서는 DTD의 존재 유무에 관계없이 일반적인 XML 문서들

```
<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>
```

그림1. XML문서 예제

을 저장하기 위하여 XML 문서를 문서 응용 영역별로 나누어 generic 스키마를 생성, 저장한다. 분할 모델방식에서의 단점인 노드 정보 취합 과정을 줄이기 위하여 XML 문서들은 하나의 단일 테이블에 노드별로 나누어 저장한다. 이때 테이블 내의 노드들의 물리적 순서가 문서내 노드의 논리적 순서와 일치하도록 설계하였으며(B-tree). 노드의 위치정보를 위하여 offset을 사용하지 않고 노드 추가, 삭제시 변경이 필요 없는 id(position id)를 사용하였다. 또한 특정노드들의 자식노드 순서정보를 직접적으로 테이블 내에 유지하지 않고 테이블 내 유효한 튜플들을 count하여 구하도록 설계하였다.

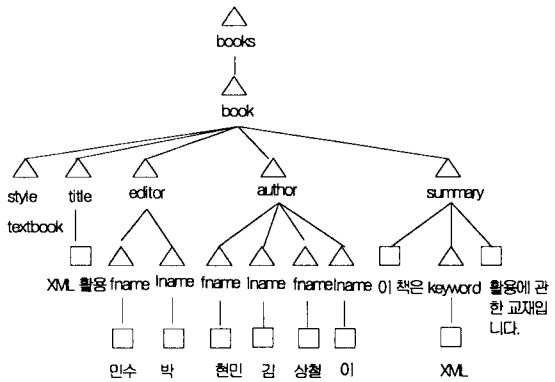

그림 2. XML 문서 예제 (그림1)에 대한 트리 표현

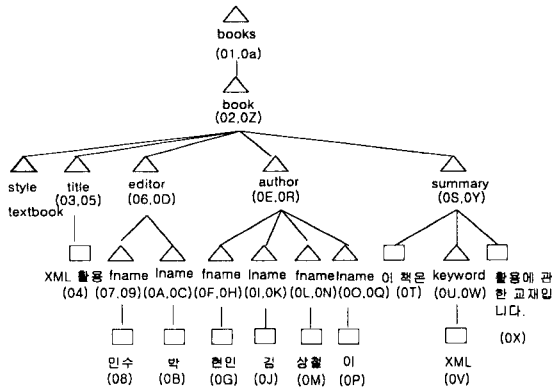

그림 3. 예제 XML 문서 트리(그림2)에 position id가 부여된 트리

<그림1>의 예제 XML 문서에 대한 트리구조는 <그림2>에 해당한다. <그림3>은 <그림2>의 트리내 각 노드별 position id를 부여한 트리 형태로서 이 때 문서내 노드의 논리적 순서와 일치하게 번호를 부여한다. 속성노드는 제외하고 엘리먼트 노드는 시작 태그와 종단 태그에 대해 각각 부여한다.

3.2 Position id

본 연구에서 제안하는 position id는 특정 노드의 위치정보를 위하여 사용하며 문서 테이블(DT)내 특정노드에 해당하는 튜플을 유일하게 식별하는 id 역할을 수행한다. 또한 삽입, 삭제와 관계없이 문서내 노드의 논리적 위치값과 테이블내 해당 튜플 id 값이 갖는 논리적 위치값이 항상 같아지는 특성을 갖고 있다.

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. 문서 저장시 연속적으로 position id가 부여된 노드사이에 XML 문서 조각의 삽입이 발생할 경우 해당 노드들에 대해 문서내의 논리적 순서와 일치되도록 연속적으로 번호를 부여한다. iid는 1부터 부여되며 0으로 끝날 수 없다. 삽입시 부여되는 pid의 규칙은 다음과 같다.

f: 삽입 노드의 선행 pid, b: 후행 pid라 할 때,

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 (62진법 사용, 단 끝이 0으로 끝나면 +1)

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

<position id 부여 예>

초기	time 1	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.

1) ST(collection)

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

2) DT(document)

docid	pid	type	content
number(4)	varchar2(305)	number(1)	varchar2(4000)

3) LT(element list)

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

4) PT(path)

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

5) ET(element)

docid	pathid	pid1	pid2	ppid1
number(4)	number(4)	varchar2(305)	varchar2(305)	varchar2(305)

6) AT(attribute)

docid	pathid	attvalue	pid1	pid2
number(4)	number(4)	varchar2(200)	varchar2(305)	varchar2(305)

7) CT(content)

docid	pathid	content	pid
number(4)	number(4)	varchar2(4000)	varchar2(305)

3.4 XML 문서의 스키마 저장 예

제안한 스키마를 이용하여 <그림1>의 문서를 저장하였을 때 실제 저장된 테이블의 데이터는 다음과 같다.

1) ST

docid	docname	description
1	books.xml	

2) DT

docid	pid	type	content
1	01	1	<books>
1	02	2	<book style="textbook">
1	03	1	<title>
1	04	9	XML 활용
1	05	3	</title>
1	06	1	<editor>
1	07	1	<fname>
1	08	9	민수
1	09	3	</fname>
1	0A	1	<lname>
1	0B	9	박
1	0C	3	</lname>
⋮	⋮	⋮	⋮
1	0a	3	</books>

3) LT

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

4) PT

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

5) ET

docid	pathid	pid1	pid2	ppid1
1	1	01	0a	
1	2	02	0Z	01
1	4	03	05	02
1	5	06	0D	02
1	6	07	09	06
1	7	0A	0C	06

1	8	0E	0R	02
1	9	0F	0H	0E
1	10	0I	0K	0E
1	9	0L	0N	0E
1	10	0O	0Q	0E
1	11	0S	0Y	02
1	12	0U	0W	0S

6) AT

docid	pathid	attvalue	pid1	pid2
1	3	textbook	02	0Z

7) CT

docid	pathid	content	pid
1	4	XML 활용	04
1	6	민수	08
1	7	박	0B

1	9	현민	0G
1	10	김	0J
1	9	상철	0M
1	10	이	0P
1	11	이 책은	0T
1	12	XML	0V
1	11	활용에 관한 교재입니다.	0X

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 자식중 n번 자식과 동일 엘리먼트명의 형제중 m번 자식은 부모노드의 위치값 즉 ppid1을 기준으로 count하여 계산, 처리한다. XQL에 의해 주어진 특정 노드의 삭제는 타 노드의 position id에 영향을 미치지 않기 때문에 갱신사항은 발생하지 않고 다만 삭제된 엘리먼트 노드를 참조하는 노드의 부모 노드 위치값 즉 ppid1 값만 조정해 주면 된다.

4. 결 론

본 연구에서는 관계형 데이터베이스를 이용하여 XML 문서를 효과적으로 검색 및 갱신을 수행할 수 있는 XML 문서 저장 시스템을 제안, 설계하였다. 빠른 검색을 위해 XML 문서를 단일 테이블에 분할 저장하였고 속성, 구조, 내용 색인을 지원하였다. 특히 노드 추가나 삭제시 노드의 위치 정보(offset) 갱신 발생을 피하기 위하여 position id를 제안 설계하였다. 향후 연구 과제로는 과다한 삭제, 삽입후 길어진 pid의 효과적인 재조정에 관한 연구가 필요하다. .

참고문헌

[1] 이규철외 4명, "효율적 XML 문서 변경 및 검색을 위한 페이징 기법", 정보과학회 학술발표논문집(I), 제26권 2호, pp 99-101, 1999

[2] 연재원, 조정수, 이강찬, 이규철. "XML 문서 구조검색을 위한 저장 시스템 설계", 정보과학회 학술발표논문집(B), 제26권 1호, pp3-5. 1999

[3] 유재수외 3명, "XML 문서에 대한 효율적인 구조 기반 검색을 위한 색인 모델", 정보과학회 학술발표 논문집, 제27권 2호, pp 18-20, 2000

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

[5] Tuong Dao, " An Indexing Model for Structured Documents to Support Queries on Content, Structure, and Attributes", Proceedings of ADL '98, pp88-97, 1998