객체-관계형 데이타베이스 기반의 XML 문서 저장 기법

(A Storage Technique for XML Documents based on Object-Relational Database)

김 지 심 † 이 기 호 ††
(Ji-Sim Kim) (Kiho Lee)

요 약 XML이 인터넷의 표준언어로 자리잡고 있음에 따라 XML을 중심으로 한 많은 데이타 관리 기술이 개발되고 있다. 특히 XML 문서를 저장하는 방식에 대한 연구가 활발히 진행되고 있는데, 이러한 기존 저장 기법들에 대해 동일한 기준으로 성능을 평가하여 효율적인 저장 기법을 제안하는 것이 필요하다. 본 논문에서는 객체-관계형 데이타베이스 모델을 이용해 XML 문서를 효율적으로 저장할 수 있는 새로운 저장 기법을 제안한다. 그리고 저장 기법들에 대한 성능 평가를 통해, 제안된 저장 기법을 효과적인 저장 기법으로 제시한다. 본 논문의 의의는 기존의 데이타 관리 모델을 사용하여 XML 문서를 보다 쉽게 효율적으로 저장할 수 있는 저장 기법을 제안하였으며, 동일한 기준을 사용한 성능 평가를 통해 XML 문서의 저장 기법들에 대한 성능을 평가하였다는 점이다.

키워드 : BFST, XML 저장 기법, 성능평가

Abstract As XML has been proposed a standard format for organizing and exchanging data in the internet, many applications on managing XML data have been developed. Especially, there are many studies for storing XML documents. It is important to evaluate the performance for efficient storage techniques. In this work, we suggest an efficient technique for storing XML documents using an object-relational database model. We verify the efficiency of a new storage technique through the performance evaluation on XML storage techniques. The contributions of this paper is that we suggest an efficient storage technique using an existing data management model and evaluate the performance for storage techniques for XML documents including an new storage technique.

Key words : BFST, XML storage, performance evaluation

1. 서 론

XML(eXtensible Markup Language) 형식으로 데이타를 자유로이 기술하고 교환하는 것이 가능해짐에 따라, 인터넷상의 애플리케이션들을 쉽게 통합할 수 있게 되었다[1]. 이에 따라, XML을 중심으로 많은 응용 연구들이 진행되고 있다. XML이 데이타를 정의하는 메타 언어라는 점을 고려할 때, 데이타 관리 및 통합 분야의 연구는 타 애플리케이션 개발의 기반이 된다고 할 수 있다. 특히, XML 문서의 저장·관리 방식은 XML 데이타의 통합 및 관리나 검색 엔진의 개발에 큰 영향을 미칠 수 있다. 그러므로, XML 문서를 효과적으로 저장하기 위한 연구들이 진행되고 있다[2].

그러나 기존의 연구들은 XML 문서로부터 추출된 스키마를 저장 시에 정확히 표현하지 못하여 문서의 정보를 일부 손실하거나, 표현이 가능하더라도 이로 인해 저장의 비효율성을 초래하는 등의 제한점을 가지고 있다. 이로 인해, XML 문서 고유의 질의 기능을 지원하지 못한다.

따라서, 본 논문에서는 BFST(Breadth First STorage)라는 새로운 저장 기법을 제안한다. 이 저장 기법은 문서로부터 추출된 정보를 손실하지 않고 저장하기 위해 객체 지향(object-oriented) 개념을 적용하고, BFS

† 비 회 원 : 이화여자대학교 컴퓨터학과
macbeal0@hotmail.com
†† 종신회원 : 이화여자대학교 컴퓨터학과 교수
khlee@ewha.ac.kr
논문접수 : 2001년 7월 26일
심사완료 : 2002년 5월 8일

(Breadth First Search) 방식으로 XML 문서를 저장한다. 그리고 저장 기법에 따른 문서의 질의 비용을 평가 기준으로 하여 성능 평가를 통해 BFST 기법의 질의 성능을 기존 연구들의 질의 성능과 비교·분석하여 BFST 기법의 효율성을 검증한다.

2. 관련연구

XML 문서를 저장하는 기법에 관한 연구들은 크게 기존의 데이타 관리 모델을 사용하는 형태와 새로운 XML 문서 전용의 데이타 서버를 개발하는 두 가지 형태로 나눌 수 있다.

관계형 데이타베이스를 기반으로 한 대표적인 시스템들로는 AT&T 연구실의 STORED를 이용한 XML 문서 저장 기법[3]과 Daniela Florescu의 애트리뷰트 인라인 저장 방식[4], XML-DBMS[5], 위스콘신 대학의 혼합 인라인 저장 기법[6] 등이 있다.

STORED를 이용한 저장 방식은 [Wang and Li][3]의 데이타 마이닝 알고리즘을 변형하여 적용한 STORED(Semistructured TO Relational Data)라는 질의 언어를 이용해 XML 문서를 관계형 데이타베이스와 오버플로우 그래프(overflow graph)에 저장한다[3]. Daniela Florescu의 논문에서는 애트리뷰트와 컨텐트를 표현하는 방식에 따라 저장 방식을 6가지로 분류하였다[4]. 이중 애트리뷰트 인라인 방식은 엘리먼트 타입별로 릴레이션을 생성하여 컨텐트와 문서의 구조 정보를 저장하는 방식으로 가장 좋은 성능을 보이고 있다. XML-DBMS는 문서의 DTD로부터 객체 지향 기반의 매핑 방식을 사용하여 관계형 데이타베이스의 스키마로 변환한다[5].

확장 관계형 데이타베이스 모델을 이용한 XML 문서 저장에 대한 시도 역시 활발하다. Oracle에서는 XML SQL Utility와 XSQL Servlet 툴을 이용해 XML 문서를 확장된 관계형 데이타베이스에 저장할 수 있고, 데이타베이스에서의 질의 결과를 다시 XML 문서로 재구성해내는 능력을 가지고 있다[7]. 그러나 구조화되어 있는 XML 문서만을 전제로 하고 있어 다양한 XML 문서들을 원하는 형태로 데이타베이스에 저장하기는 힘들며, BLOB으로 저장되어 있는 일부분은 다시 풀텍스트(full-text) 기반의 검색을 행해야 한다는 점에서 오버헤드가 존재한다.

또한, 객체 지향 데이타베이스인 O2를 이용한 Lore[8], eXcelon[9], POET CMS[10], 바다 IV[11]와 같은 객체 지향 데이타베이스를 기반으로 한 XML 서버 등이 있다. 이러한 연구들은 XML 문서의 계층화된 구조와 객체 기반 데이타베이스가 잘 호환될 수 있어 XML 문서 모델의 무결성을 유지할 수 있으며, 저장이나 검색을 효율적으로 수행할 수 있는 장점을 가지고 있다. 그러나 일반적으로 XML 문서의 각 엘리먼트들은 매우 작은 단위들로 나뉘어져 있고, 이러한 엘리먼트들이 데이타베이스의 각 객체와 매핑될 때 공간을 불필요하게 많이 차지할 수 있다. 그리고 객체 지향 기반의 저장 시에는 XML 문서의 DTD가 변경될 경우 데이타베이스의 스키마를 다시 변경해주어야 하는 어려움이 있다. 이에 비해, 본 연구에서는 DTD가 변경되어도 스키마를 쉽게 변경할 수 있다.

본 논문은 관계형 데이타베이스를 이용해 XML 문서를 저장하는 데에 초점을 두고 있다. 이는 인터넷의 많은 데이타들이 이미 관계형 데이타베이스를 기반으로 구축되어 있으며, 가장 일반적으로 사용하는 관계형 데이타베이스를 기반으로 XML 문서를 보다 쉽게 효율적으로 저장할 수 있기 때문이다.

표 1은 관계형 데이타베이스 기반의 XML 저장 기법에 관한 연구들에서 공통적으로 제안한 고려 사항과, XML-QL, XQL 등의 XML 질의 언어에서 제안한 중요 질의 유형을 기준으로 하여 관련 연구들의 특징을 비교·분석한 것이다. XML의 구성요소를 저장하지 못하는 경우는 ×로 표현되었으며, 저장을 할 수 있으나 저장 형태가 비효율적이거나 질의시 질의 시간이 오래 걸리는 등의 문제를 초래하는 경우에는 △로 표현하였다.

표 1 관계형 데이타베이스 기반의 XML 저장 기법 비교

고려 요소		STORED	애트리뷰트인라인	XML-DBMS	혼합 인라인
문서 정보	계층 구조	×	△	△	△
	순서정보	△	○	○	×
	동적 타입의 엘리먼트	△	○	×	×
저장 형태	집합값애트리뷰트	△	○	△	△
	데이타의 정규화	×	○	○	×
XML질의유형	구조 정보 질의	×	×	×	×
	경로 표현	×	×	×	○

표 1에서와 같이 대부분의 기존 연구들은 추출한 XML 문서의 고유 정보를 효율적으로 저장하지 못하고 있다. 예를 들어, STORED를 이용한 저장 방식에서는 동적 타입의 엘리먼트인 ANY형으로 선언된 데이타가 추가되는 경우 오버플로우 그래프에 저장해 저장의 비일관성을 초래하며, XML-DBMS에서는 ANY 형의 엘리먼트를 전혀 저장하지 못한다. 혼합 인라인 방식에서는

ANY 형으로 선언된 동적인 엘리먼트는 저장은 되지만, 질의 시에 추가적으로 비용이 든다. 그리고, XML 문서의 고유한 특징 중의 하나인 집합값 애트리뷰트[6]의 경우, 기존의 관계형 데이타베이스에서 효과적인 해결책을 지원하지 못하므로, 기존 연구들에서는 다른 릴레이션에 분리하여 저장하므로, 질의 시간이 증가하는 문제점을 가지고 있다. 또한, 데이타가 정규화되지 않고 중복된 데이타가 여러 릴레이션에 걸쳐 나타나기도 한다.

XML의 대표적인 질의 언어인 XML-QL이나 XQL에서는 공통적으로 정규 경로 표현을 중요한 질의 형태로 제시하고 있다. 그러나 기존 연구들에서는 XML 문서의 구조 정보를 질의하지 못하고, 직접적인 정규 경로 표현이 불가능하여 조인으로 대치하고 있다. 그러므로, 관련 연구들은 비효율적인 XML 문서의 저장형태로 인하여 고유한 XML 질의 유형 기능을 지원하지 못하고 있다. 본 연구에서는 이러한 문제점들을 보완하여 XML 문서를 효율적으로 저장하고 XML 문서의 여러 질의 유형들을 지원할 수 있도록 설계하였다.

3. BFST(Breadth First STorage) 기법을 이용한 저장 시스템 설계

본 연구에서는 기존 연구들의 제한점을 극복할 수 있는 BFST 기법을 제안한다. BFST 기법에서는 표 1의 기존 연구의 문제점을 다음과 같은 방법으로 해결하여, 저장공간을 감소하고 질의 비용을 줄일 수 있게 하였다.

1) 문서 정보

· 계층(hierarchy) 구조 : XML 문서가 계층적 구조로 정의되어 있으므로, 경로 표현(regular path expression)을 이용하여 문서를 탐색(navigate)·질의할 수 있어야 한다. 따라서, 본 연구에서는 계층 정보에 대한 질의를 위해 객체 지향 개념을 적용하여 OID(Object IDentifier)를 통한 참조로 문서의 계층 구조를 표현하였다.

· 동적 타입의 엘리먼트 : 선택적 엘리먼트나 ANY 타입으로 선언된 엘리먼트들은 DTD에만 나타나거나, 인스턴스에만 나타나거나, 혹은 DTD와 인스턴스 모두에 나타날 수 있는 요소들이다. 그러므로, 이를 정확히 저장하기 위해서는 DTD와 인스턴스를 모두 분석한다.

2) 저장 형태

· 집합값 애트리뷰트(set-valued attribute) : 현재 집합값 애트리뷰트는 관계형 데이타베이스로는 해결할 수 없다. 그러므로, 본 논문에서는 객체-관계형 데이타베이스 모델을 이용하여 집합값 애트리뷰트들을 중첩 테이블(nested table)에 저장하여 빠르고 효과적으로 질의할 수 있도록 제안한다.

3) XML 문서 고유 질의 지원

· 구조 정보 질의 : 기존의 연구에서는 XML 문서에 고유하게 요구되는 구조 정보를 질의할 수 있는 기능이 미흡하므로, 본 연구에서는 구문 분석 후 저장을 위해 생성한 포스팅 파일을 재사용하여 문서의 구조 정보까지 질의할 수 있도록 하였다.

· 경로 표현(regular path expression) : 기존의 연구에서와 달리 본 연구에서는 이러한 형태의 질의를 지원하기 위해 객체 지향 개념을 도입하였다. OID(Object IDentifier)를 이용하여 객체들의 계층 구조를 표현할 수 있어, 기존 연구에서 조인으로 대체되던 경로 표현을 자연스러운 형태로 가능하게 하였다.

BFSTN 기법을 이용한 시스템 구성도는 그림 1과 같다.

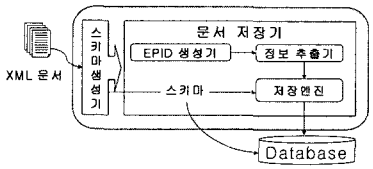

그림 1 시스템 구성도

XML 문서를 입력으로 받은 스키마 생성기는 우선 DTD의 구문을 분석하며 문서의 구조 정보를 추출하여 스키마를 생성하고, 이를 바탕으로 하여 인스턴스의 구문을 분석하여 정확한 스키마를 완성한다.

문서 저장기는 생성된 스키마를 바탕으로 XML 문서를 직접 데이타베이스에 저장한다. 문서를 저장하기 위해 저장기는 DOM으로 모델링된 XML 문서의 엘리먼트와 애트리뷰트에 EPID(Element Path IDentifier)를 부여하게 되고, 부여된 EPID를 기반으로 저장에 필요한 구조 정보 및 내용 정보를 추출해 내어 이 정보들을 포스팅 파일에 작성한다. 저장 엔진에서는 BFS 방식으로 순회하며 문서를 데이타베이스에 저장하게 된다.

3.1 스키마 생성기

본 논문에서는 그림 2의 유효한(Valid) 문서인 conference.xml를 사용하였다.

BFST 저장 기법에서는 XML 문서의 DTD와 인스턴스를 모두 분석하여 스키마를 생성한다. 우선, DTD를 분석하여 기본적인 계층 관계에 있는 엘리먼트들과 애트리뷰트들은 객체 테이블과 해당 테이블의 필드로 생성한다. DTD를 기반으로 생성된 스키마 테이블을 이용하여 인스턴스를 분석하여 선택적 애트리뷰트, 집합값 애트리뷰트, ANY 형의 엘리먼트 등 인스턴스에서만 동적으로 나타나는 엘리먼트들을 분석하여 스키마를 완성한다.

```
<!ELEMENT conference ( inproceedings* | proceedings* ) >
<!ELEMENT inproceedings ( inpro_book, url? ) >
<!ATTLIST inproceedings key CDATA #REQUIRED >
<!ATTLIST inproceedings crossref IDREF #IMPLIED >
<!ELEMENT inpro_book ( author* | title | pages | cdrom | year | crossref | booktitle | ee)>
<!ELEMENT author (#PCDATA) >
<!ELEMENT title (#PCDATA) >
<!ELEMENT pages (#PCDATA) >
<!ELEMENT cdrom (#PCDATA) >
<!ELEMENT year (#PCDATA) >
```

```
<!ELEMENT crossref (#PCDATA) >
<!ELEMENT booktitle (#PCDATA) >
<!ELEMENT ee (#PCDATA) >
<!ELEMENT url (#PCDATA) >
<!ELEMENT crossref (#PCDATA) >
<!ELEMENT proceedings ( pro_book, url? ) >
<!ATTLIST proceedings key ID #REQUIRED >
<!ELEMENT pro_book ( editor* | title | booktitle | series | volume | publisher| year | isbn)>
<!ELEMENT editor (#PCDATA) >
<!ELEMENT series ( href? | source? ) >
<!ELEMENT href (#PCDATA) >
<!ELEMENT source (#PCDATA) >
<!ELEMENT volume (#PCDATA) >
<!ELEMENT publisher (#PCDATA) >
<!ELEMENT isbn (#PCDATA) >
<conference>
<inproceedings key="AbowdWM95">
<inpro_book>
<author>Gregory D. Abowd</author>
<author>Hung-Ming Wang</author>
<author>Andrew F. Monk</author>
<title>A Formal Technique for Automated Dialogue Development.</title>
<pages>219-226</pages>
...
```

그림 2 예제 문서(conference.xml)

conference.xml에서 생성한 스키마의 일부는 표 2와 같다.

inpro_book, conference와 같이 자식 엘리먼트들을 하나 이상 가지는 엘리먼트들은 객체 테이블로 분류되고, 이러한 객체 테이블로 선언된 엘리먼트의 자식 엘리먼트들을 각 객체테이블의 속성을 이룬다. 데이타 타입은 각 엘리먼트의 텍스트를 검토하여 선언한다. 또한, 연속적으로 두 번 이상 등장하는 author, editor와 같은 집합값 애트리뷰트는 중첩 테이블로 선언되어 객체 테이블 내에 함께 저장되므로 질의 비용을 줄일 수 있다.

3.2 문서 저장기

저장기 모듈에서는 XML 문서를 구문 분석하고 'conference'를 루트로 갖는 DOM으로 재구성하여 DOM 트리의 모든 엘리먼트와 애트리뷰트에 EPID를 부여한다. 그림 3에서 doc 타입은 루트와 동일하므로 생략하였다. 만약 <?XML ENCODING='UTF-8'?>와 같은 PI 옵션가 존재할 경우, 루트 노드의 자식 노드로 표현될 수 있으나, EPID는 부여되지 않는다. EPID는 엘리먼트들의 경로, 관계 그리고 순서 등의 정보를 담고 있는 식별자이다. 이를 기반으로 저장 시에 엘리먼트들의 관계를 유지하여 저장할 수 있다. conference.xml의 DOM 트리에는 그림 3과 같이 EPID가 부여된다.

각 EPID는 각 엘리먼트의 부모 엘리먼트의 id와 해당 레벨 내에서의 해당 엘리먼트의 순서 정보를 포함하

표 2 Conference.xml의 객체-관계형 스키마

테이블형식	테이블 이름	객체 타입	속성	데이타 타입
객체테이블	inpro_book	inpro_book_type	EPID	varchar(20)
			author	author_set
			title	varchar(270)
			pages	varchar(270)
			cdrom	varchar(270)
			year	integer
			crossref	varchar(270)
			booktitle	carchar(270)
			ee	varchar(270)
	conference	conference_type	EPID	varchar(20)
			inproceedings	ref inproceedings_type
			proceedings	ref proceedings_type
	inproceedings	inproceedings_type	EPID	varchar(20)
			key	varchar(270)
			crossref	varchar(270)
			inpro_book	ref inpro_book_type
			url	varchar(270)
중첩테이블	author	author_type	EPID	varchar(20)
			author	varchar(270)
	editor	editor_type	EPID	varchar(20)
			editor	varchar(270)

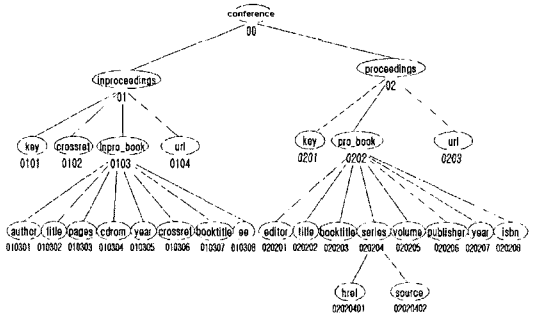

그림 3 EPID가 부여된 DOM 트리

고 있다. 이렇게 EPID가 부여된 DOM 트리를 BFS 방식으로 순회하며 우선, 자식 엘리먼트를 가지는 부모 엘리먼트와 컨텐트들을 상위 객체 테이블로 저장한다. 그리고 자식 엘리먼트들을 종속 객체 타입으로 정의하여 테이블에 저장한다. 저장 형태의 일부는 그림 4와 같다.

3.3 구현환경

마이크로소프트사의 WindowsNT 4.0 상에서 Oracle8i를 이용해 XML 문서를 저장하였다. 그리고 BFST 기법을 오라클 상에서 Java로 구현하기 위해 오라클 사에서 제공하는 JDBC 2.0을 사용하였다. XML 문서의 구문 분석은 JavaCC를 사용하여 작성된 Silfide XML Parser로 수행하였으며, 웹 상에서 XML 문서를 선택하고 저장하기 위해 사용자 인터페이스는 Java Servlets를 사용하여 구현하였다.

그림 5는 XML 문서를 선택하여 입력한 경우 결과 화면을 보여주고 있으며, 그림 6은 질의 결과를 보여주고 있다.

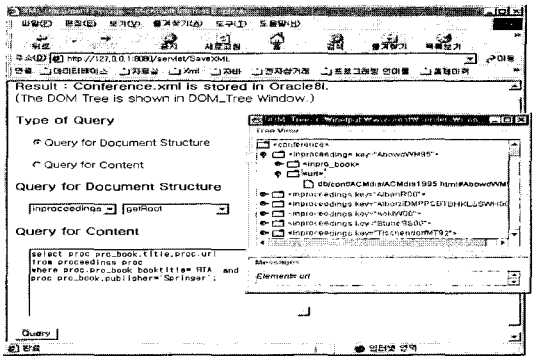

그림 5 저장 결과

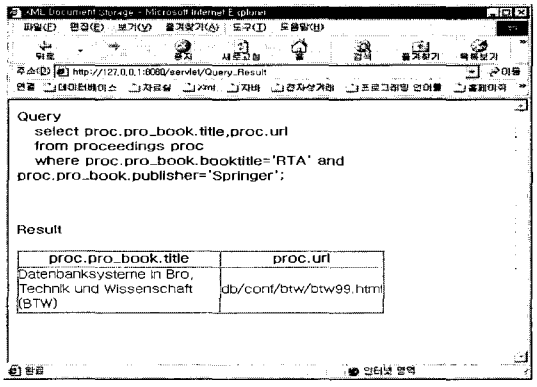

그림 6 질의 결과

4. 저장 기법에 따른 성능 평가

4.1 성능 평가 환경

본 논문에서는 XML문서의 구조와 크기에 따라 성능

<inproceedings>

EPID	key	crossref	inpro_book	url
01	AbowdWM95	NULL	ref(inpro_book)	http://conf/ACMdis/ACMdis1995.htm#AbowdWm95.htm
02	AlbenR00	NULL	ref(inpro_book)	http://conf/ACMdis/ACMdis2000.htm#AlbenR00.htm
03	AlborziDMPPSBTBHKLS..	NULL	ref(inpro_book)	db/conf/ACMdis/ACMdis2000.html#AlborziDMPPSBTBHKLSSWH00.htm

<inpro_book>

EPID	author		title	pages	cdrom	year	cross-ref	book title	ee
	EPID	author							
0102	010201	Gragory D.Abowd	A Formal Technique for Automated Dialogue...	219-226	NULL	1995	NULL	Symposium on Designing Interactive Systems	http://www.acm.org/pubs/citations/...
	010202	Hung-Ming Wang							
0202	020201	Lauralee Alben	Envisioning the E-Quarium: Strategic Design Planning...	452-454	NULL	2000	NULL	Symposium on Designing Interactive Systems	NULL
	020202	Michael Rigsby							

그림 4 BFST 기법의 저장 형태

평가를 실시하였다. 첫 번째는 10Mbytes의 Movie.xml 문서를 대상으로 실험하였다. 두 번째 성능 평가는 예제 문서인 Conference.xml 문서를 평가 대상으로 한다. 이 문서는 30Mbytes의 크기로 되어 있다. 각 문서는 동일한 구조를 가진 여러 문서들이 하나의 DTD를 공유하는 형태로 이루어져 있으며 Oracle에 저장된다.

또한, 본 성능 평가는 BFST 기법과 함께 기존의 저장 기법들 중에서 애트리뷰트 인라인 방식과 XML-DBMS의 두 가지 저장 방식을 채택하여 성능 평가를 수행하고 결과를 비교·분석하였다.

4.2 성능 평가 기준

본 연구에서는 여러 질의 비용 기준 중에서 시간 비용을 평가 기준으로 삼는다. XML 질의 유형에 대한 분류를 바탕으로 성능 평가에 적용할 예제 질의들을 선정하고 질의 시간을 측정하여 성능 평가를 수행하였다. 표 3과 같이 각 질의 유형을 선정하였고, 오라클 상에서 질의문을 적용하여 결과가 출력되기까지의 질의 시간을 측정하였다.

표 3의 질의 유형은 P. Griffiths Selinger가 [12]에서 제시한 관계형 데이타베이스의 질의 유형 분류와 [13]에서의 XML-QL, XQL에서의 XML 질의 유형을 기반으로 하여 XML 문서에서 가능한 모든 질의 유형을 좀 더 상세히 분류한 것이다.

순환 기반의 질의(Traversal-based Query)는 XML 문서의 엘리먼트 경로를 탐색하여 질의하고자 하는 데이타를 찾는 경로 표현을 사용한 질의를 말한다. 이 때 conference. inproceedings과 같이 경로를 명시적으로 언급하는 것은 간단한 경로 표현(Simple Path Expression)

표 3 성능평가를 위한 질의 유형

질의 유형	연산
Predicate-based Query	column = content
	column>content
	column between content1 and content2
	column IN (list of contents)
	column IN sub-query
	Or
	And
	Not
Traversal-based Query	Simple Path Expression
	Arbitrary Path Expression
Link-based Query	One-document Link
	Multi-document Link
Sort	
Grouping	
Join	

으로 분류하였다. 그리고 '*.editor'나 '*.author'와 같이 XML-QL에서 경로가 불분명하여 와일드카드(*)를 사용한 정규 표현으로 나타낸 질의는 임의의 경로 표현(Arbitrary Path Expression)으로 분류하였다.

각 질의 유형에 따라 Conference.xml 문서에 적용될 수 있는 질의 예제는 표 4와 같다.

표 4의 각 예제 질의들을 BFST 방식에서 SQL로 형식화하여 표현하면 다음의 표 5와 같다.

4.3 성능 평가 결과 및 분석

여기에서는 30Mbyte 문서 대상의 성능 평가의 결과를 종합하여 표 6으로 제시한다.

표 4 선정된 질의 예제(Conference.xml)

질의 유형	예제 질의	
Predicate-based Query	*Q1.* Rakesh Agrawal이 쓴 논문의 제목을 찾아라.	
	Q2. 1996년 이후의 Stefano Ceri가 쓴 XML에 관한 논문의 제목을 찾아라.	
	Q3. 1998년~2000년 사이에 나온 XML에 관한 논문의 연도, 제목, 저자를 찾아라.	
	Q4. 시리즈 중에서 'Advances in Data Base Theory'를 소스로 가진 책의 제목을 찾아라.	
Traversal-based Query	Simple Path Expression	*Q5.* db/conf/popl/popl82.html#Cooper82인 url에 등장하는 책의 제목을 찾아라.
		Q6. key가 AfonsoRS99인 inpro_book의 모든 사항을 출력하라.
	Arbitrary Path Expression	*Q7.* *.editor=Gerard Salton의 논문을 찾아라.
Link-based Query	One-document Link	*Q8.* conf/icdt/86 키를 가진 proceedings를 참조하는 inpro_book의 title을 찾아라.
	Multi-document Link	*Q9.* /conf/er/99w 키를 가지고 있는 proceedings를 참조하는 inproceeding의 url을 모두 찾아라.
Sort	*Q10.* pro_book 논문의 제목을 연도별로 정렬하여 출력하라.	
	Q11. Serge Abiteboul이 쓴 논문의 제목을 연도별로 출력하라.	
Grouping Query	*Q12.* 논문 편수가 최대인 저자의 이름과 논문 편수를 출력하라.	
	Q13. 저자별 논문수를 세어 출력하라.	
Join	*Q14.* Boniier91키를 가진 inproceedings의 key, 제목, url을 찾아라.	

표 5 SQL 문으로 표현된 형식 질의

예제	BFST방식에서의 형식 질의
Q1.	Select i.title From inpro_book i,TABLE(i.author) a Where a.author='Rakesh Agrawal';
Q2.	Select i.title From inpro_book i,TABLE(i.author) a Where i.year>=1996 and a.author='Stefano Ceri';
Q3.	Select i.year,i.title,a.author From inpro_book i,TABLE(i.author) a Where i.year BETWEEN 2001 AND 2002 and title LIKE '%XML%';
Q4.	Select proc.title From proceedings proc Where proc.pro_book.series.source='Advances in Data base Theory';
Q5.	Select i.inpro_book.title From inproceedings i Where i.url='db/conf/popl/popl82.html#Cooper82';
Q6.	Select i.inpro_book.author,i.inpro_book.title,i.inpro_book.pages,i.inpro_book.year, i.inpro_book.booktitle,i.inpro_book.ee From inproceedings i Where i.key='AfonsoRS99';
Q7.	Select inp.i_title,pro.p_title From (select a.author aut,i.title i_title from inpro_book i,TABLE(i.author) a where a.author='Gerard Salton') inp, (select e.editor edi,p.title p_title from pro_book p,TABLE(p.editor) e where e.editor='Gerard Salton') pro Where inp.i_title=pro.p_title(+);
Q8.	Select i.title From inproceedings inproc,inpro_book i,proceedings proc Where proc.key='conf/icdt/86' and proc.key=inproc.crossref;
Q9.	Select i.url From inproceedings i,proceedings p Where p.key='conf/er/99w' and i.crossref=p.key;
Q10.	Select p.year,p.title From pro_book p Order by p.year;
Q11.	Select i.year,i.title From inpro_book i,TABLE(i.author) a Where a.author='Serge Abiteboul' Order by i.year;
Q12.	Select editor_count.ed_name,max_table.max_title From (select max(title_num) max_title from (select e.editor ed_name, count(p.title) title_num from pro_book p,TABLE(p.editor) e group by e.editor) editor_num) max_table, (select e.editor ed_name, count(p.title) title_num from pro_book p,TABLE(p.editor) e group by e.editor) editor_count Where max_table.max_title=editor_count.title_num(+);
Q13.	Select a.author,count(i.title) From inpro_book i,TABLE(i.author) a Group by a.author;
Q14.	Select inproc.key,inproc.inpro_book.title,inproc.url From inproceedings inproc Where inproc.key='Bonnier91';

표 6 질의 유형에 따른 예제 질의

(단위 : ms)

질의 유형		예제 질의	BFST 기법	애트리뷰트 인라인 방식	XML-DBMS
Predicate-based Query		*Q1.*	3.2	15.6	18.1
		Q2.	2.3	16.2	8.8
		Q3.	7.9	25	17.8
		Q4.	0.5	6.9	×
Traversal-based Query	Simple Path Expression	*Q5.*	3.2	19.4	9.3
		Q6.	8.5	90.2	43.1
	Arbitrary Path Expression	*Q7.*	2.1	20.5	15.3
Link-based Query	One-document Link	*Q8.*	5.0	6.1	12.9
	Multi-document Link	*Q9.*	10.4	16.5	7.1
Sort		*Q10.*	4.0	26.7	13.3
		Q11.	3.4	23	16.8
Grouping Query		*Q12.*	0.4	12.9	0.1
		Q13.	3.4	14.9	2.6
		Q14.	3.0	57.4	36.5

이를 그래프로 비교하면, 그림 7과 같다.

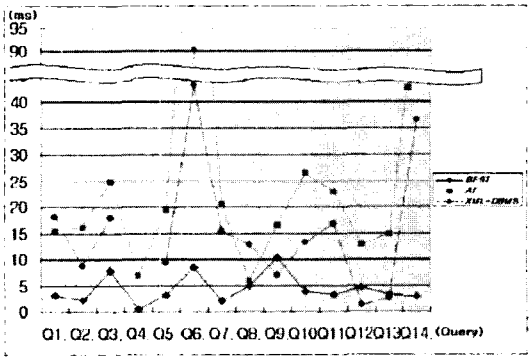

그림 7 성능 평가 비교 그래프

전반적으로 BFST 기법이 다른 저장 기법들에 비해 질의 시간이 더 빠른 것으로 나타났다. 그리고 애트리뷰트 인라인 방식이 가장 느린 질의 시간을 보였는데, 이는 엘리먼트 타입 수만큼 릴레이션이 생성되고, 각 릴레이션들은 문서의 논리적 구조와는 상관없이 엘리먼트의 타입별로 데이타들을 가지고 있기 때문이다. 따라서, 동일한 엘리먼트 타입을 가진 컨텐트를 질의하는 경우에는 하나의 릴레이션만을 읽기 때문에 질의 시간이 빠르다. 그러나, 다른 엘리먼트 타입 간의 데이타들을 질의할 때에는 질의 문장이 훨씬 복잡해지고 조인의 수가 급격하게 증가하므로, 질의 시간이 매우 느려진다.

또한, XML-DBMS에서는 각 릴레이션들은 직접적으로 부모/자식 관계에 있는 컨텐트들을 저장하고 있으며, 이것은 BFST 기법과 유사한 형태이다. 그러나 조상/자손 관계 등에 있는 엘리먼트들은 다른 릴레이션들에 분리되어 저장된다. 즉, BFST 기법에서는 어떤 엘리먼트이든지 원래 문서의 계층 구조를 유지하면서 저장된다. 그러나 XML-DBMS에서는 엘리먼트들은 구조와 상관없이 분리된 릴레이션들로 존재하며 단지 기본키/외래키에 의해서만 관계가 명시되어 있으므로, 원래 문서의 계층적 구조 정보를 손실하였다. 이로 인한 차이는 하나의 테이블 내에서 질의할 때보다 여러 테이블에 걸쳐 질의할 때에 더욱 뚜렷하게 나타난다. BFST 기법에서는 계층 구조 관계에 있는 엘리먼트들은 OID를 통한 참조로 표현하였으므로, 포인터로 다른 테이블의 데이타를 직접 찾을 수 있다. 그러므로, XML-DBMS에서는 기본키/외래키를 이용해 여러 릴레이션을 조인하여 질의를 느리게 수행하는 반면, BFST 기법에서는 여러 테이블에 걸친 데이타를 OID를 통해 훨씬 빠르게 질의를 할 수 있다.

또한, XML-DBMS에서 부모/자식 관계에 있는 엘리먼트라도 Conference.xml의 author나 editor와 같은 집합값 애트리뷰트의 경우에는 inproceedings나 proceedings 테이블과는 별도의 다른 릴레이션들에 분리되어 저장되어 있다. 따라서, Q1 등 집합값 애트리뷰트에 대한 질의 역시 조인을 통해 질의가 이루어지게 되고 질의 시간이 지연된다. 그러나 BFST 기법에서는 집합값 애트리뷰트인 author나 editor 엘리먼트의 경우 부모 엘리먼트인 inproceedings나 proceedings 엘리먼트가 생성한 릴레이션 내에 중첩 릴레이션 형태로 함께 저장되기 때문에, 저장 공간을 줄일 수 있고 질의 시에도 조인을 통해서가 아니라 데이타를 직접 가지고 올 수 있게 되므로 질의 시간을 감소시킬 수 있다.

또한, 질의의 표현에 있어서도 BFST 기법이 XML 문서에서 고유하게 요구하는 질의를 더욱 효과적으로 표현할 수 있다는 것을 알 수 있다. XML-QL에서는

XML 질의 언어의 요구 조건 중 하나로 conference.inproceedings.inpro_book.title과 같이 간단한 경로 표현을 이용해 컨텐트를 찾을 수 있어야 한다고 명시하였다. 그러나 경로 표현을 이용한 질의는 관계형 데이타베이스에서는 직접적으로 표현이 불가능하다. 따라서, 애트리뷰트 인라인 방식이나 XML-DBMS에서는 조인을 이용한 질의문으로 변환하여 간접적으로 표현해야 하며, 이로 인해 질의 시간도 느려짐을 알 수 있다. 그러나 BFST 기법에서는 위에서 언급한 Q5~Q7의 질의문과 같이 임의의 길이로 된 경로 표현을 이용해 사용자가 원하는 대로 질의문으로 작성할 수 있고, OID를 이용해 데이타를 직접 찾아오므로 질의 시간을 감소시킬 수 있었다. Q4 예제에서 볼 수 있듯이, XML-DBMS에서는 ANY 타입의 엘리먼트와 같이 인스턴스에서 동적으로 나타나는 엘리먼트의 경우에 저장되지 않았으므로 질의가 불가능하다. 이는 XML-DBMS가 DTD로부터 스키마를 추출하여 문서를 저장하기 때문이며, 따라서 ANY 형의 엘리먼트와 같이 인스턴스에만 나타나는 엘리먼트들에 대한 정보를 손실하는 경우가 발생한다.

따라서, 본 성능 평가에서도 알 수 있듯이 성능 평가 결과, 본 논문에서 제안한 BFST기법이 XML 문서의 질의를 빠르게 지원하는 효율적인 저장 기법임을 알 수 있다.

이 외에도, 하나의 DTD를 XML 문서들이 공유할 경우에는 스키마 생성기에서의 분석을 통해 스키마를 통합할 수 있고, 멀티 DTD를 가지는 XML 문서들의 경우 객체 관계형 데이타베이스의 'ALTER'와 같은 스키마 변경 명령을 통해 구조를 변경할 수 있다.

저장된 XML 문서들은 데이타나 구조의 삽입, 삭제 등은 스키마와 문서 저장기에서의 추가적인 분석을 통해 관리할 수 있다. 이후에도 데이타베이스 내에서 XML 문서를 관리할 수 있는 방법들은 데이타베이스에서 기존에 제시하고 있는 툴들을 통해 해결할 수 있으며, 추가적인 연구들은 향후 과제로 연구될 수 있을 것이다.

본 논문은 XML 형태로 인터넷의 데이타들이 구축되어 있는 경우에 XML 문서를 저장하여 관리하는 데에 초점을 맞추고 있다. 특히, XML 문서를 질의·검색하기 위해 대용량의 XML 문서를 저장하는 데에 중점을 두고 있다. 또한, EDI나 전자 상거래에서는 문서의 구조나 형식은 동일하면서 내용만 달라지며 이러한 문서들에 대한 작업이 반복되는 경우가 많으므로, 같은 DTD를 공유한 대용량의 문서들을 BFST 기법으로 데이타베이스에 저장한다면 문서의 검색이나 질의를 훨씬 쉽고 빠르게 수행할 수 있다.

5. 결 론

본 논문에서는 XML 문서를 효과적으로 저장할 수 있는 새로운 저장 기법을 제안하였다. BFST 기법을 이용하면, 기존의 연구들이 저장 시에 XML 문서의 정보를 일부 손실하거나, XML 문서의 질의 유형을 효과적으로 지원하지 못하는 등의 한계점들을 해결하여 XML 문서를 저장할 수 있다.

본 논문의 의의는 크게 다음과 같이 두 가지로 요약될 수 있다.

첫 번째, 본 논문에서 제시한 BFST 기법은 XML 문서를 질의하기 위한 검색 시스템은 물론, 인터넷의 XML 기반 지식 관리 시스템에서 이용될 수 있다. XML 문서의 속성을 잃지 않고 원래의 문서와 똑같은 형태를 유지하여 저장할 수 있는 BFST 기법을 이용한 시스템을 사용한다면, 웹에서 XML 문서를 효과적으로 저장하고 관리할 수 있다.

두 번째로, 본 논문에서 수행한 저장 기법에 따른 질의 성능 평가는 여러 저장 시스템들에 대한 성능 평가에 활용될 수 있다. 본 논문에서 적용한 XML 질의 유형의 분류는 XML-QL 및 XQL에서 공통으로 제시한 것을 바탕으로 분류한 것이며, XML 문서에서 필수적으로 행해져야 하는 질의 유형들을 분류해 놓은 것이다. 이를 바탕으로 하여 본 논문에서 수행한 것과 같이 성능 평가가 이루어지고 각 저장 기법들의 장·단점을 분석할 수 있다면, 이는 여러 저장 기법들의 효율성을 검증하는 데에 유효하게 사용될 수 있을 것이다.

참 고 문 헌

[1] 추기능, XML의 활용 및 표준화 동향, 정보통신정책연구원, p.10, http://www.xmlgo.net/, 2000.

[2] Serge Abiteboul, Peter Buneman and Dan Suciu, Data on the Web-From Relations to Semistructured Data and XML, p.258, Morgan Kaufmann, 1999.

[3] Dan Suciu, Alin Deutsch and Mary Fernandez, "Storing Semi-structured Data Using {STORED}," in ACM SIGMOD, pp.431-442, 1999.

[4] Daniela Florescu and Donald Kossmann, "A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database," INRIA Technical Report, INRIA, No. 3680, 1999.

[5] Ronald Bourret, XML-DBMS Version 1.01, http://www.rpbourret.com/xmldbms/readme.htm, 2000.

[6] Jayavel Shanmugasundaram, Kristin Tufte, etc. "Relational Databases for Querying XML Docu-

ments:Limitations and Opprotunities," in Proceedings of the 25th VLDB Conference, pp.302-04, 1999.
[7] Oracle Corporation, XML Support in Oracle8i and Beyond, An Oracle Technical Whitepaper, http://technet.oracle.com/tech/xml/info/index2.htm?Info&htdocs/xml_twp.html, 1998.
[8] F.Bancihon, G. Barbedette, "The design and implementation of O2, an object-oriented Database System," In proceedings of the second international workshop on object-oriented database, pp.3-32, 1998.
[9] Larry. R. DeBoever, eXcelon B2B Integration Server, eXcelon Corp, 1997.
[10] SAN MATEO, POET Software and Staffware Partner to Provide Streamlined XML Content Solutions, California, 1999.
[11] 이명철, 장동준, "바다-IV/XML 검색기 설계 및 구현", Korean Database Conference 2000, pp.288-296, 2000.
[12] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie and T. G. Price, "Access Path Selection in a Relational Database Management System," in Proceedings of ACM SIGMOD, pp.82-93, 1979.
[13] 장경자, "XML 질의 언어를 SQL로 변환시키는 번역기", 이화여자대학교 석사학위 청구논문, 1999.

김 지 심
1993년 부산 학산여자고등학교 졸업. 1997년 이화여자대학교 유아교육학과 졸업. 1999년 연세대학교 전산원 SE과정 수료. 2001년 이화여자대학교 과학기술대학원 졸업. 현재 명지대, 명지전문대 출강

이 기 호
정보과학회논문지 : 컴퓨팅의 실제
제 8 권 제 1 호 참조