버전을 지원하는 XML 저장관리 시스템 설계 및 구현

손 충 범[†] · 오 경 근[††] · 유 재 수[†††]

요 약

최근 인터넷을 이용한 전자문서의 중요성이 부각되면서 대용량의 XML 문서에 대해 효율적으로 저장하고, 검색하며, 관리할 수 있는 XML 저장관리 시스템의 연구가 활발히 진행되고 있다. XML 응용에서 특허 문서 관리, 소프트웨어 설계, 시스템 매뉴얼 등의 응용과 같이 수정된 기존의 문서들이 관리되어야 하는 분야에서 버전 관리 기능이 필요하다. 본 논문에서는 문서의 수정을 효율적으로 지원하는 분할모델을 이용하여 문서 수정에 따른 버전닝을 지원하는 데이터 모델을 제안한다. 또한 버전닝을 지원하는 XML 저장관리 시스템을 설계하고 구현한다. 구현한 시스템이 기존 저장 시스템 보다 더 우수하다는 것을 성능평가를 통해 보인다.

Design and implementation of an XML Repository System supporting Document Version

Chung Beom Son[†] · Kyoung Keun Oh[††] · Jae Soo Yoo[†††]

ABSTRACT

Recently, as the importance of the management on internet documents has highly increased, the research of an XML repository system has been actively made to store, retrieve and manage large XML documents. The version management for XML documents is required in the XML applications such as patent documents, software design and system manual that the modified documents have to be managed. In this paper, we propose a data model based on a fragmentation model that supports document versioning. We also design and implement an XML repository system supporting document versioning. It is shown through performance evaluation that our system outperforms the existing repository system.

키워드 : XML, 저장관리 시스템(Repository System), 버전관리(Version Management), 데이터 모델(Data Model)

1. 서 론

월드와이드웹 컨소시움에 의해 제안된 XML(eXtensible Markup Language)은 HTML과 SGML의 장점을 수용하여 문서의 논리적인 구조를 표현하면서도 쉽게 사용할 수 있도록 만든 것으로 1998년 W3C에서 XML을 권고안으로 채택하였다. 현재 인터넷 웹 문서, 전자도서관, CSCW, EDI, EC, CALS 등을 포함한 다양한 분야, 수학 분야의 MathML, 채널 기술의 CDF, 무선인터넷에서의 WML 등에서 XML 연구는 상당히 활발하게 진행되고 있으며 이러한 분야에서의 많은 양의 XML 문서를 효율적으로 저장하고, 검색하며, 관리할 수 있는 시스템의 필요성이 점차 증대되고 있다[1].

이와 같이 XML을 활용한 다양한 응용분야에 적용되는 정보 시스템을 구축하기 위해 가장 중요한 저장관리 시스템(repository system)의 기능은 XML 문서에 대한 정보검색과 정보관리를 위한 것이다[4]. 이는 XML로 표현된 구조화된 정보의 특성을 반영하여 가장 적절하게 구현하고 기능을 제공하는 시스템으로 XML의 특징인 정보 구조화의 패러다임을 그대로 유지시킬 수 있는 정보검색이 가능하여야 한다. 즉, 엘리먼트 단위의 검색뿐만 아니라 엘리먼트 간의 논리적인 계층 구조를 이용한 검색 등을 수용할 수 있어야 한다. 또한 XML 문서에 표현할 수 있는 모든 정보에 대한 사용자의 요구에 부응하기 위해서는 XML 문서 내에 포함된 모든 정보를 손실 없이 저장하고 수정하며 관리 할 수 있어야 한다. 즉, 일반 텍스트 문서에 비해 XML 문서는 내용 정보뿐만 아니라 문서의 논리적인 구성을 기술하는 구조정보를 가지고 있기 때문에 이에 대한 대응이 필요하며, XML 문서가 가지고 있는 내용은 단순한 텍스트에서 다양한 멀티미디어를 포함할 수 있다. 이러한 XML 문서는 구조정보를 이용하여 문서를 효율적으로 관리하기 때문에 데이터베이스에 XML 문서, DTD, 구조정보 및 다양한 미디어를 저장하고 관리되어야 한다[1]. 또한 XML 문서에 대한 버전관리는 특허 문서 관리, 소프트웨어 설계, 시스템 매뉴얼 등의 응용과 같이 수정된 기존의 문서들이 관리되어야 하는 분야에서 절

※ 본 연구는 학술진흥재단 선도연구자사업(KRF-2001-041-E00233)의 지원으로 수행되었음.
† 준 회 원 : 충북대학교 대학원 정보통신공학과
†† 정 회 원 : 가인정보기술연구소 연구원
††† 종신회원 : 충북대학교 정보통신공학과 교수
논문접수 : 2002년 4월 3일, 심사완료 : 2002년 10월 1일

실한 기능이지만 대부분의 시스템들은 이에 적절히 대응하지 못하고 있다.

본 논문에서는 문서의 수정을 효율적으로 반영할 수 있는 분할모델을 이용하여 문서 수정에 따른 버전을 지원하는 데이터 모델을 제안한다. 또한 내용변경과 구조변경에 따른 버전닝 방법을 제시하고 제안한 버전 기반의 XML 저장관리 시스템을 설계하고 구현한다. 또한 테스트를 통해 설계한 버전을 지원하는 XML 저장관리 시스템을 검증한다.

본 논문의 구성은 다음과 같다. 2장에서 XML 저장관리 시스템의 기존 연구들을 살펴보고, 3장에서는 XML 저장관리 시스템의 요구사항과 설계한 구조 표현 방법 및 버전닝 방법을 설명한다. 4장에서는 구현한 XML 저장관리 시스템에 대해 기술하며, 5장에서 XML 저장관리 시스템의 성능평가 결과를 보여주고, 마지막으로 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

대용량의 XML 문서에 대한 효율적인 저장, 관리 및 검색을 지원하는 XML 저장관리 시스템은 많은 정보 시스템 응용 분야에서 반드시 필요하다. 기존에 XML 저장관리 시스템의 연구는 XML 문서의 저장방식에 따라 연구를 크게 구분하면 분할모델과 비분할 모델로 구분할 수 있다. 분할 모델이란 XML 문서를 저장할 때 문서를 구성하고 있는 엘리먼트를 나누어 저장하는 방법으로 문서의 수정에 대해서는 효율적인 반면 해당 문서에 대한 검색이 발생한 경우 엘리먼트들을 재구성하여 검색 결과를 반환하는 과정에서 시스템의 성능을 저하시키는 문제가 발생한다. 비분할 모델은 XML 문서 전체를 저장한 후 각각의 엘리먼트는 위치정보를 가지고 접근하는 방식이다. 이는 문서를 한꺼번에 저장하였기 때문에 통합 과정이 필요 없어 문서 참조를 빨리 할 수 있지만, 내용의 일부만이 수정되었을 때도 문서 전체를 재구성해야 한다는 단점이 있다.

현재 대부분의 DBMS 회사들이 XML 문서를 수용할 수 있게 확장하고 있다. 오라클사의 오라클 9i 서버와 함께 사용되는 오라클 XML DB는 XML Type과 XML Repository를 제공하는데, XML 데이터를 컬럼의 데이터타입으로 저장할 수 있게 XML Type을 지원하고, XML Repository는 XML 데이터 및 문서의 관리를 위한 인터넷 저장소를 제공한다. DB2의 XML extender[14]에서는 XML 문서를 여러 테이블로 저장하거나 CLOB 또는 BLOB 형태로 저장한다. 또한 XML 전용 DBMS인 소프트웨어 AG의 tamino[13]는 컬렉션 형태로 여러 개의 문서 형태로 구성되어 관리되고, ObjectStore를 확장해서 만든 eXcelon[16]은 XML 문서를 개별적인 XML 엘리먼트로써 객체들로 저장한다. 이들 제품들은 디스크 기반에서 XML 문서를 관리하지만, 실시간 환경에서 XML 데이터를 효율적으로 관리하기 위한 주기억장치 기반의 XML 저장관리기에 대한 연구가 진행중에 있다[17].

XML 문서의 버전 관리에 관한 연구를 살펴보면 다음과 같다. RCS(Revision Control System)[2]와 같은 기존의 문서 버전관리 시스템은 라인 중심적이고 많은 제약과 성능 문제를 가지고 있다. RCS는 가장 최신의 버전을 완전히 저장하는 반면에 다른 모든 버전들은 역방향 편집 스크립트로서 저장된다. 이 스크립트들은 문서의 개발 이력에서 어떻게 역방향으로 가야하는지를 기술한다. 최신 버전을 제외한 다른 버전을 위해 구버전을 생성하기 위해 역방향 편집 스크립트를 적용해야 하는 추가의 처리과정을 필요로 한다. RCS는 원본 문서의 논리적 구조를 유지하지 않기 때문에 XML 문서의 구조 검색을 어렵게 만들고 버전을 재구성하는데 비용이 많이 들게 된다.

편집기반의 UBCC(Usefulness Based Copy Control)[3]는 편집에 바탕을 둔 방법들을 향상시켰다. UBCC는 주어진 버전의 유효한 객체들을 데이터 페이지들에서 군집화한다. 이것은 페이지 유용성의 개념을 사용한다. 한 페이지에서 유효한 많은 객체들이 어떤 임계값 이하면 유효한 모든 페이지 객체를 다른 페이지로 복사되어진다. 하나의 버전은 그 다음에 유용한 페이지들을 단지 접근함으로써 재구성되어 진다. 특정 문서의 버전을 재구성하는 것은 버전을 포함하는 객체들을 찾아서 그것들을 정확한 논리적 순서로 생성하는 것과 같다. 그러나 버전들의 편집에 바탕을 둔 표현은 문서 내용을 재배열한다든지 문서를 재구성하는 변경에 대해선 효율적으로 표현할 수가 없다. 이런 문제점을 해결하기 위해 편집스크립트를 제거하고 일반 세그먼트의 개념을 사용하는 복사에 바탕을 둔 버전닝 방법인 복사기반의 UBCC[3]가 나왔다. 이 방법은 편집기반의 UBCC에서 사용한 유용성을 기본으로 하는 페이지 관리를 확장하여 각 버전을 편집 스크립트없는 부리스트들의 리스트를 구현함으로써 저장 및 검색을 수행한다. 이 복사 기반의 방법은 변경없는 세그먼트 레코드들에 대해 편집 스크립트를 분리해 가질 필요없이 리스트로 객체들을 저장할 수 있으며 그 리스트들 중간에서 변경이 생기면 단지 분할만 하면 된다. 그리고 문서 재구성과 같은 변경을 융통성 있게 처리할 수 있으며 버전 검색 I/O 오버헤드를 줄일 수 있다.

3. XML 저장관리 시스템 설계

3.1 고려 사항

최근 다양한 형태의 문서들이 XML로 전자 문서화하여 관리하려는 연구가 전자상거래, 전자도서관, 전자 정부, 기술 문서 관리 등의 다양한 응용 분야에서 급격히 증대되고 있으며, 이렇듯 XML을 활용한 다양한 응용분야에서 적용되는 정보시스템을 구축하기 위해서 가장 중요한 기능이 정보검색과 정보관리를 위한 저장소(Repository) 기능이다. 이는 XML로

표현된 구조화된 정보의 특성을 반영하여 가장 적절하게 구현하고 기능을 제공하는 시스템으로 XML 문서에 표현할 수 있는 모든 정보에 대한 사용자의 요구에 부응하기 위해서는 모든 정보의 손실 없이 저장하고 수정하며 관리 할 수 있어야 하며, 특히 XML의 특징인 정보 구조화의 패러다임을 그대로 유지시킬 수 있는 정보검색이 가능하여야 한다.

이러한 특성을 갖는 XML 저장관리 시스템을 개발하는데 있어서 반드시 고려해야 될 사항으로는 XML 문서 모델링으로 XML 문서를 저장하고 관리하기 위한 데이터 모델은 저장 시스템에 삽입하기 전의 문서와 저장된 문서를 다시 복원했을 때의 데이터 손실을 방지하기 위해 XML의 모든 특성들을 고려해야 한다. 즉, XML 문서는 복잡한 구조와 다양한 멀티미디어 데이터를 포함할 수 있다. 이러한 XML 문서는 구조 정보를 이용하여 문서를 효율적으로 관리하기 때문에 데이터베이스에 XML 문서, DTD, 구조정보 및 다양한 미디어를 저장, 관리할 수 있어야 하며 특히, 저장된 XML 문서에 대한 수정이 효율적으로 이루어질 수 있도록 데이터 모델이 설계되어야 한다. 또한 특정 DTD에 의존하는 데이터 모델링이 아닌 다양한 DTD를 수용할 수 있어야 한다. 또한 구조 정보 검색 기능으로 XML 문서는 엘리먼트들의 집합으로서 각 엘리먼트 간의 계층적 및 수평적 관계성을 포함하는 특성을 갖고 있다. 이에 XML 문서에 대한 검색은 기존 시스템의 문서 단위 검색 이외에 임의 깊이에서 나타나는 엘리먼트 단위 검색이 가능해야 한다. 특히 XML 문서에 내재된 논리적인 구조에 기반한 검색으로써 엘리먼트들의 계층 간의 관계, 같은 계층 내에서의 관계 등을 고려하여야 한다. 계층간의 관계는 부모/자식/조상/자손 관계가 있으며, 같은 계층내의 관계는 형제 엘리먼트간의 순서가 있다. 또한 엘리먼트에 나타날 수 있는 속성에 질의로 애트리뷰트 이름과 값을 주고 해당하는 문서나 엘리먼트를 찾는 검색을 지원해야 한다. XML 문서에 대한 검색 결과는 질의를 만족하는 XML 문서이거나 혹은 XML 문서의 특정 일부분이 될 수 있다. 문서의 일부분은 엘리먼트 혹은 엘리먼트를 구성하는 애트리뷰트나 내용의 일부일 수 있다. 또한 XML 문서의 히스토리 정보를 중요시하는 응용분야를 위해 버전(Version) 정보를 관리할 수 있어야 한다. XML 문서에 대한 버전은 특허 문서 관리, 소프트웨어 설계, 시스템 매뉴얼 등의 응용과 같이 수정된 기존의 문서들이 관리되어야 하는 분야에서 적절히 활용될 수 있다.

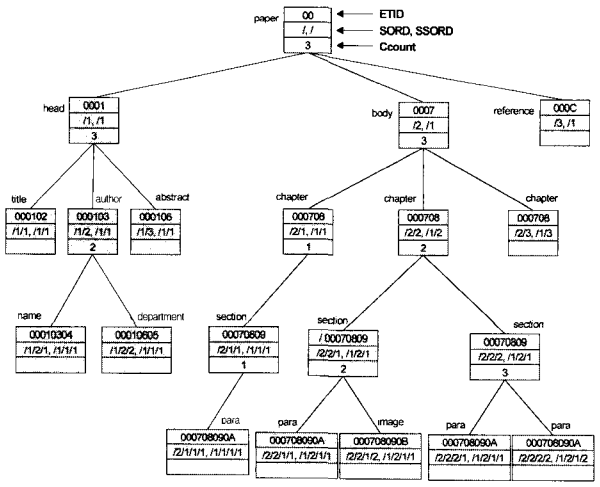
이 이외에 사용자 권한에 의한 액세스 제어 기능, 다수 저자의 동일 문서에 대한 공동 저작시 발생하는 문서의 불일치를 방지하는 *check-in/check-out* 기능 등 부가적인 기능들을 제공해야 한다.

3.2 구조정보 표현 방법

XML 문서에 대해 문서의 논리적 구조 기반의 검색을 지원하기 위해서는 문서의 논리적 구조 정보를 표현하는 방법이 제시되어야 한다. 본 논문에서 구조정보는 엘리먼트 이름을 식별할 수 있는 ID, 부모와 자식 엘리먼트간의 계층정보, 동일한 부모 엘리먼트를 갖는 자식 엘리먼트(이하 형제 엘리먼트)들의 순서정보, 그리고 동일한 부모 엘리먼트를 갖는 자식들 중 동일한 타입의 엘리먼트들에 대한 순서정보로 표현된다. 이들은 부모 엘리먼트의 정보를 유지한다. 그러므로 기준 엘리먼트로부터 특정 엘리먼트에 대한 계층정보와 순서정보를 간단한 문자열 조작만으로 쉽게 구할 수 있다. 이렇게 구한 순서정보는 각 엘리먼트에 유일하게 할당된 값이기 때문에 직접 특정 엘리먼트를 접근할 수 있다. 엘리먼트들 간의 계층 정보를 표현하기 위해서는 엘리먼트 이름에 대해 식별할 수 있는 *EID(Element ID)*를 부여해야 한다. 이러한 EID는 DTD에서 정의된 각 엘리먼트에 대하여 유일한 ID를 의미한다. 각 엘리먼트에 대해 ID를 부여하는 방식을 통해서 DTD의 논리적 구조를 분석할 때 발생하는 엘리먼트간의 순환 문제와 ANY로 설정된 엘리먼트 처리 문제를 제거할 수 있다.

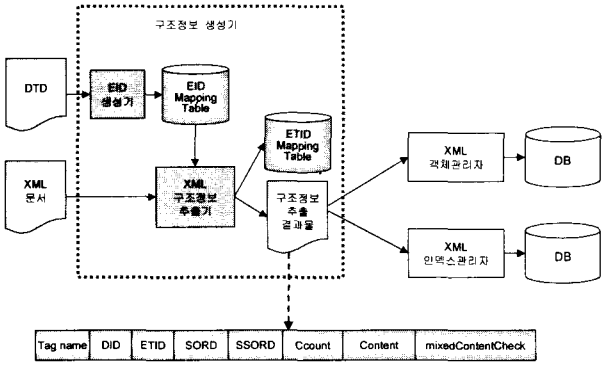
본 논문에서는 특정 엘리먼트를 구별하면서 엘리먼트들 간의 계층정보를 표현하기 위해 EID를 사용하여 문서상의 엘리먼트들에 ETID(Element Type ID)를 부여한다. 이렇게 부여된 ETID는 문서의 논리적 구조를 나타내게 된다. 또한 XML 문서에서는 DTD에 나타난 발생지시자에 의한 반복적인 엘리먼트의 사용이 가능하다. 이렇게 반복적으로 사용된 엘리먼트들은 ETID만으로는 구별이 불가능하다. 따라서 두 종류의 순서정보, 형제 엘리먼트들의 발생순서를 나타내는 SORD(Sibling ORDer)와 동일한 타입의 형제 엘리먼트들 간의 순서정보를 나타내는 SSORD(Same Sibling ORDer)가 그것이다. 또한 특정 엘리먼트의 자식 검색을 위해 엘리먼트가 갖는 자식의 수를 Ccount를 사용하여 유지한다.

(그림 1)에서는 간단한 XML 문서를 제안한 구조정보인 ETID, SORD, SSORD, Ccount를 사용해서 트리의 형태로 표현한 예를 보여 준다.



(그림 1) XML 문서의 구조 정보

XML 문서의 구조정보는 XML 구조정보 추출기와 EID (Elemet ID) 생성기에 의해서 얻어진다. EID 생성기는 DTD로부터 각각의 엘리먼트들에 대해 엘리먼트 타입을 추출하고 엘리먼트 이름과 EID를 사상시켜 주는 매핑 테이블을 구성한다. 그리고 XML 구조정보 추출기는 XML 문서를 분석하면서 EID 매핑 테이블을 참조하여 ETID, SORD, SSORD를 생성하고 Ccount를 계산하여 문서의 구조정보를 추출한다. 이렇게 추출된 문서의 구조정보는 곧바로 XML 객체 관리기와 XML 인덱스 관리기에게 넘겨지고 각 관리기에서는 필요한 정보만을 선택적으로 데이터베이스에 저장한다. 이 과정을 그림으로 표현하면 (그림 2)와 같다.



(그림 2) 구조정보추출기 구성도

3.3 스키마 설계

본 논문에서 설계한 XML 데이터 모델은 (그림 3)과 같다. DTD 테이블은 저장할 XML 문서의 DTD를 저장하고 관리한다. 엘리먼트 테이블은 XML 문서를 분할하여 각 엘리먼트별로 저장하고 관리한다. 멀티미디어 테이블은 XML 문서 내에 포함된 멀티미디어를 저장하고 관리한다. 버전 테이블은 XML 문서에 대한 버전닝이 발생할 경우 가장 마지막에 버전된 문서의 정보를 유지하는 테이블이다. 마지막으로 DOC 테이블는 XML 문서에 대한 파일명을 유지하는 테이블이다.

3.4 버전 기능

기존의 문서와는 다른 구조화된 특성을 갖는 XML 문서는 내용만을 삽입하거나 삭제할 수도 있지만 문서의 구조가 변하면서 내용이 추가되거나 삭제되는 경우도 있다. 그렇기 때문에 XML 문서에 대한 수정은 크게 내용 변경과 구조 변경을 동시에 고려하여야만 한다.

이에 본 논문에서는 버전 관리를 크게 두 가지 즉, 문서의 구조는 변하지 않고 단지 내용만이 수정되는 경우와 문서의 구조가 바뀌면서 내용이 수정되는 경우로 구분한다. 내용 변경시에는 내용 부분에 대해서만 버전을 유지하는 엘리먼트 단위 버전을 제공하고, 구조 변경시에는 구조정보가 달라지기 때문에 문서 단위 버전을 제공한다.

DTD table

DTDID	DTDNAME	CONTENT
1	Paper.dtd	< !ELEMENT paper (head, body, reference +) > < !ATTLIST paper status (public | confidential) "public" > .....
		.

ELEMENT table

DID	SORD	DID_history	STR_history	ELE_history	CONTENT	mixed-ContentCheck	Ccount
1	/	1	1	1	< ? xml version = "1.0" encoding = "euc - kr"? > < ! DOCTYPE paper SYSTEM paper.dtd > < paper status = "public" > < /paper >	0	3
1	/1	1	1	1	< head > < /head >	0	3
1	/3	1	1	1	< reference > 참고문헌 < /reference >	0	0
1	/2/2	1	1	1	< chapter > 2. 색인구조 본 장에서는… < /2/2/1 > < /2/2/2 > < /chapter >	1	2

MULTIMEDIA table

DID	NAME	SORD	DID_history	STR_history	ELE_history	CONTENT
1	fig1.gif	/2/2/1/2	1	1	1	

VERSION table

DID	lastDID_history	lastStructureVersion
1	1	1

DOC table

DID	DTDID	NAME
1	1	강형일.xml

(그림 3) XML 데이터 모델링

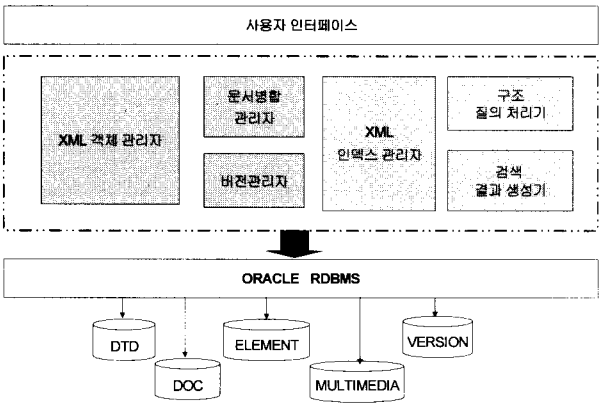
4. XML 저장관리 시스템 구현

4.1 시스템 구성도

구현한 XML 저장관리 시스템은 자바를 사용하여 유닉스 환경에 RDBMS로 오라클 8.0.1을 사용하였으며, 웹 환경에서 지원하도록 아파치 웹서버 1.3.9와 서블릿과 JSP를 지원하는 톰캣 3.0을 사용하였다.

(그림 4)는 구현한 XML 저장관리 시스템 구조를 보여준다. 구현한 XML 저장관리 시스템 구조를 보여준다. 구현한 XML 저장관리 시스템은 XML 객체관리자, XML 인덱스 관리자, 버전관리자, 문서병합관리자, 구조질의 처리기, 검색결과 생성기로 구성된다. XML 저장관리 시스템을 구성하는 각 모듈의 역할은 다음과 같다.

XML 저장관리 시스템에서 가장 핵심적인 기능을 담당하는 XML 객체관리자에서는 실제 XML 문서를 저장하기 위한 스키마 생성 및 XML 문서 인스턴스의 저장 및 추출을 담당한다. XML 버전관리자는 XML 문서에 대한 버전관리를 담당하고, XML 문서병합 관리자는 요청한 문서에 대한 병합을 담당한다.

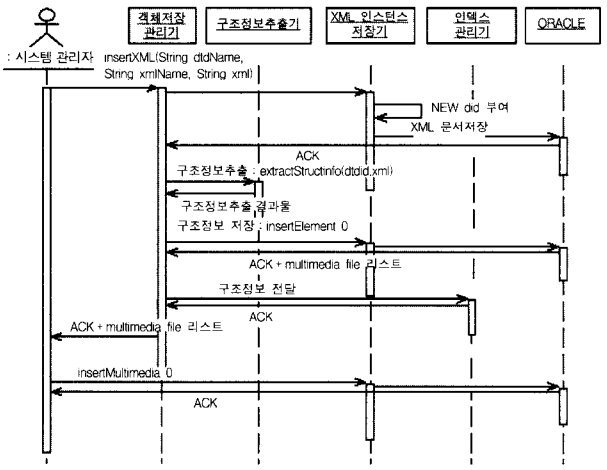


(그림 4) 저장 관리 시스템 구성도

XML 인덱스 관리자에서는 내용 검색만을 위한 내용 색인기, 구조 검색을 지원하는 구조 색인기, 애트리뷰트 검색을 지원하기 위한 애트리뷰트 색인기로 구성된다. 구조질의 처리기는 XML 문서에 대한 다양한 검색을 처리하기 위한 모듈로 실제 질의 처리시 각 질의 타입에 따라 SQL을 생성하는 모듈이다. 또한 검색결과 생성기에서는 구조질의 처리기에서 처리한 검색 결과를 가지고 XML 객체관리자의 XML 인스턴스 관리기를 통하여 문서 전체 또는 일부분을 사용자에게 제공하는 역할을 수행한다.

4.2 객체 관리자

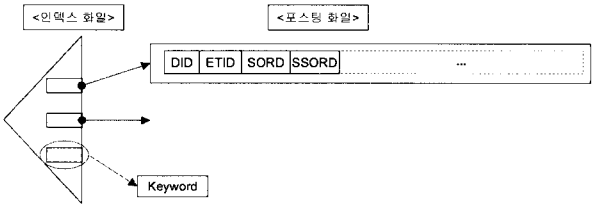
XML 객체관리자는 구조 정보를 가지는 XML 문서를 효율적으로 관리하는 기능을 담당한다. XML 객체관리자에서는 구조화된 문서를 효율적으로 관리하기 위해 문서 관리기능을 담당한다. XML 객체관리자를 구성하는 세부 모듈은 객체저장관리기, 구조정보 추출기, XML 인스턴스 저장기, XML 스키마 생성기로 이루어진다. 객체 저장관리기는 XML 객체관리자를 구성하는 각 모듈들에 대한 통합 인터페이스를 제공하고, 구조 정보추출기는 XML 문서의 구조에 기반에 검색을 제공하기 위해 구조정보를 추출하는 모듈이다. XML 인스턴스 저장기는 저장할 XML 인스턴스를 엘리먼트 단위로 엘리먼트 테이블에 저장하며 XML 인스턴스가 포함하고 있는 멀티미디어 파일을 구조 정보와 함께 멀티미디어 테이블에 저장한다. 또한, 내용 변경이나 구조 변경을 통해 생성된 버전 문서에 대해서도 저장을 실시한다. XML 스키마 생성기는 새로운 DTD에 대하여 해당 DTD를 저장하고, XML 인스턴스가 저장관리기에 의해 저장될 때 XML 인스턴스를 분할하여 저장 관리하기 위한 엘리먼트 테이블과 XML 문서에 포함된 멀티미디어 파일을 저장하기 위한 멀티미디어 테이블을 생성한다. (그림 5)는 객체관리자의 XML 문서저장 흐름도이다.



(그림 5) XML 문서 저장 흐름도

4.3 색인 관리자

XML 문서에 대한 다양한 검색을 지원하기 위해서는 XML 문서의 특성을 고려한 색인 구조가 설계되어야 한다. 색인은 앞 절에서 추출한 구조 정보를 이용하여 생성하는데 내용 색인, 구조 색인, 애트리뷰트 색인의 3개로 구성되며 기존의 역화일 방식을 사용한다.

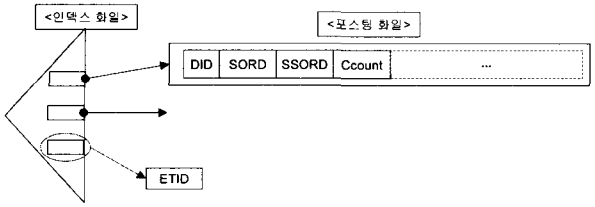


(그림 6) 내용 색인의 구조

내용 색인은 내용 검색을 지원하는 색인로서 XML 문서로부터 추출된 색인어로 구성된 색인 화일과 색인어가 출

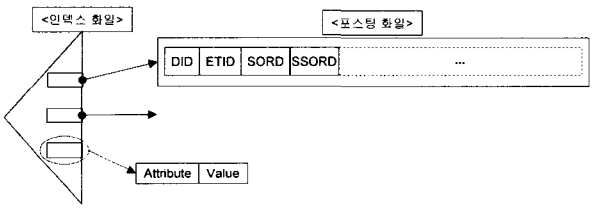
현한 문서와 엘리먼트의 정보를 나타내는 포스팅 화일로 구성된다. 포스팅 화일은 DID, VERSION, ETID, SORD, SSORD로 구성된다. DID와 VERSION은 문서 단위 검색의 경우 사용되며, ETID, SORD, SSORD는 엘리먼트 단위의 검색을 지원하기 위해 사용된다. (그림 3), (그림 7)은 내용 색인의 구조를 나타낸다.

구조 색인은 구조 검색을 지원하는 색인으로서 문서의 논리적인 구조 정보를 손실없이 표현한다. 이를 통해 엘리먼트 간의 계층관계 검색, 엘리먼트 간의 순서 관계인 형제 검색을 지원한다. (그림 7)은 구조 색인의 구조를 나타낸다.



(그림 7) 구조 색인의 구조

애트리뷰트 색인은 애트리뷰트 검색을 지원하는 색인으로서 추출된 구조 정보 중 애트리뷰트 이름과 값을 색인으로 구성하고 이와 관련된 DID, VERSION, ETID, SORD, SSORD들을 포스팅 화일로 구성한다. (그림 8)은 애트리뷰트 색인의 구조를 나타낸 것이다.



(그림 8) 애트리뷰트 색인의 구조

4.4 버전 관리자

XML 버전 관리자는 수정된 기존의 XML 문서에 대한 버전 관리를 할 수 있도록 지원하는 모듈이다. 처음 새로 생성된 XML 문서 인스턴스는 엘리먼트 테이블의 DID_history, ELE_history, STR_history 필드에 버전 1을 부여받은 이후에 문서에 대한 수정이 이루어지면 사용자는 새로운 버전에 추가할 것인지를 판단한다. XML 문서에 대한 버전 관리는 엘리먼트 테이블의 DID_history 필드에서 관리를 한다. DID_history 필드 값은 원본 XML 문서에 대한 버전이 수행될 경우 지속적으로 유지하는 버전 번호이다. XML 문서에 대한 수정이 내용만이 변경될 경우에는 엘리먼트 테이블의 ELE_history 필드에 버전 정보를 기술하여 엘리먼트 단위 버전을 지원하며, 문서의 구조가 변경되었을 경우에는 전체 XML 문서의 구조정보가 바뀌게 되므로 내용 변경과 구분하기 위해 엘리먼트 테이블의 STR_history 필드에 버전 정보를 기술한다. 이때 이전의 버전닝된 문서가 내용만이 변경된 경우에는 ELE_history 필드 값을 리셋 시킨다. 또한 사용자가 특정 문서를 요구할 때 디폴트로 마지막에 버전닝된 문서를 빠르게 추출하기 위하여 VERSION 테이블의 lastDID_history 필드 값을 DID_history 값으로 대치시킨다. 이때 문서의 구조가 변경된 경우에는 lastStructureVersion 필드에 엘리먼트 테이블의 STR_history 필드 값으로 대치시켜서 구별한다.

(그림 9)는 DID가 1인 문서에 대한 내용 변경이 발생할 경우 ELEMENT 테이블과 VERSION 테이블을 보여주고 있다. (그림 5)에서 보듯이 abstract 엘리먼트의 내용이 변경된 경우에 그 엘리먼트에 대한 버전 정보로서 DID_history와 ELE_history 정보를 1씩 증가하여 엘리먼트 테이블에 저장한다. 또한, VERSION 테이블에 lastDID_history 필드 값을 1 증가시킨다.

ELEMENT table

DID	SORD	DID_history	STR_history	ELE_history	CONTENT	mixedContentCheck	Ccount
1	/	1	1	1	< ? xml version = "1.0" encoding = "euc-kr" ? > < ! DOCTYPE paper SYSTEM paper.dtd > < paper status = "public" > < /paper >	0	3
1	/1	1	1	1	< head > < /head >	0	3
1	/2	1	1	1	< body > < /body >	0	3
1	/3	1	1	1	< reference > 참고문헌 1 < /reference >	0	0
1	/1/1	1	1	1	< title > 효율적인 구조검색을 위한 … < /title >	0	0
1	/1/2	1	1	1	< author > < /author >	0	2
1	/1/3	1	1	1	< abstract > 최근 인터넷이 … < /abstract >	0	0
1	/2/2	1	1	1	< chapter > 2. 색인구조 본 장에서는 … < /2/2/1 > < /2/2/2 > < /chapter >	1	2
1	/1/3	2	1	2	< abstract > XML 문서는 논리적인 구조 … < /abstract >	0	0

VERSION table

DID	lastDID_history	lastStructureVersion
1	2	1

1st Version 내용변경

(그림 9) 내용 변경시 테이블

ELEMENT table

DID	SORD	DID_history	STR_history	ELE_history	CONTENT	mixedContentCheck	Ccount
1	/1/3	2	1	1	< abstract > XML 문서는 논리적인 구조 … < /abstract >	0	0
1	/	3	2	1	< ? xml version = "1.0" encoding = "euc-kr" ? > < ! DOCTYPE paper SYSTEM paper.dtd > < paper status = "public" > < /paper >	0	3
1	/1	3	2	1	< head > < /head >	0	4
1	/2	3	2	1	< body > < /body >	0	3
1	/3	3	2	1	< reference > 참고문헌 1 < /reference >	0	0
1	/1/1	3	2		< title > 효율적인 구조검색을 위한 … < /title >	0	0
1	/1/2	3	2	1	< author > < /author >	0	2
1	/1/3	3	2	1	< author > < /author >	0	2
1	/1/4	3	2	1	< abstract > XML 문서는 논리적인 구조 … < /abstract >	0	0

VERSION table

DID	lastDID_history	lastStructureVersion
1	3	2

2st Version 구조변경

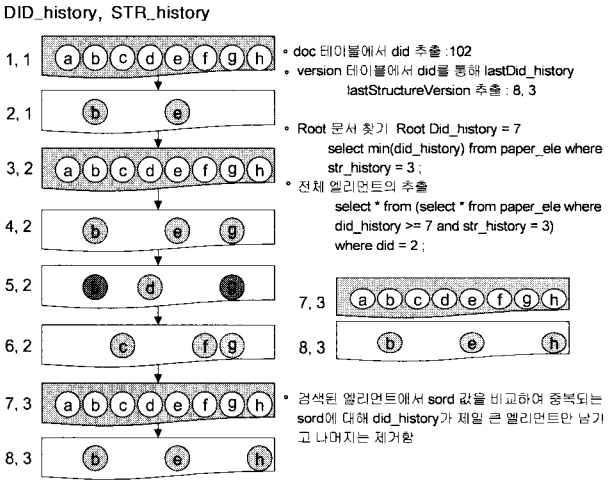
(그림 10) 구조 변경시 테이블

(그림 10)은 내용 변경된 문서가 다시 구조 변경이 되었을 때 XML 버전 관리기에 의해 변경된 테이블들의 내용을 보여주고 있다. (그림 6)에서 보듯이 XML 문서의 구조가 변경되는 경우에 구조 정보를 새롭게 추출하여 해당하는 정보를 ELEMENT 테이블에 DID_history와 STR_history 값을 1씩 증가시키고 추출된 정보들과 같이 테이블에 저장된다. 또한 VERSION 테이블에 lastDID_history와 lastStructure-Version 값을 1 증가시킨다.

4.5 문서병합 관리자

XML 문서병합 관리자는 사용자가 요구하는 문서와 문서에 포함된 멀티미디어를 데이터베이스로 부터 추출하는 기능을 담당한다. XML 문서병합 관리자는 문서 전체에 대해서는 해당 문서에 대한 DID와 DID_history나 문서 명을 통해서 추출이 된다. 문서의 일부분인 엘리먼트에 대해서는 해당 문서의 DID, DID_history 및 해당 엘리먼트의 SORD를 통해 추출한다.

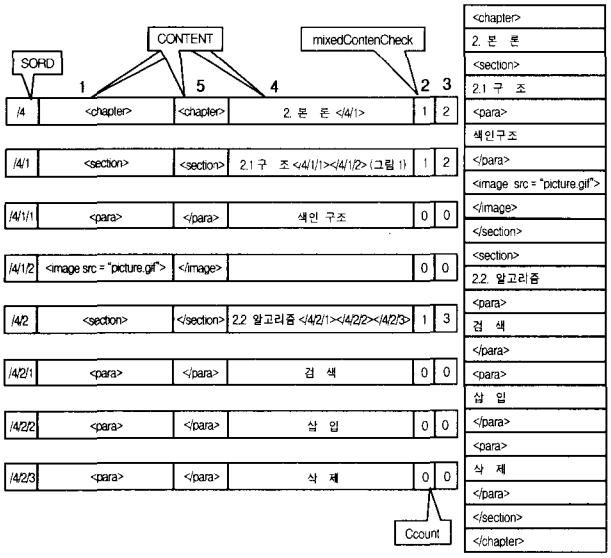
본 논문에서는 XML 문서를 각 엘리먼트별로 분할하여 저장 관리하기 때문에 문서를 추출하는 경우 먼저, 문서를 구성하는 엘리먼트들을 추출한 다음 이를 이용하여 문서를 통합하는 과정이 필요하다. 먼저, 문서를 구성하는 엘리먼트 추출시 XML 문서병합 관리자는 DID_history와 STR_history 값을 구한 다음 문서를 구성하는 전체 엘리먼트들을 추출한다. 이를 위해 가장 마지막에 구조 변경된 문서를 구성하는 엘리먼트들을 추출하고 이후에 내용 변경된 엘리먼트들이 존재한다면 이 부분의 엘리먼트들도 추출한다. 추출한 엘리먼트에서 SORD 값을 비교하여 중복되는 SORD에 대해 DID_history가 제일 큰 엘리먼트만 남기고 나머지는 제거한다. (그림 11)은 버전닝이 8번 이루어진 가운데 구조 변경이 3번 이루어진 문서 중에 마지막에 버전닝된 문서를 구성하는 엘리먼트들을 추출하는 과정을 보여주고 있다.



(그림 11) 문서를 구성하는 엘리먼트 추출 예

문서 통합 과정은 ELEMENT 테이블에서 추출한 SORD, CONTENT, mixedContentCheck, Ccount를 이용하여 문서 병합을 한다. 먼저 SORD 값으로 sorting을 하고나서 문서 병합은 다음에 방법을 이용한다. Root 엘리먼트는 XML 선언 부분 및 DTD 선언 부분과 시작 태그까지 기록한다. mixed-ContentCheck가 0이면서 Ccount 값이 없으면 끝 태그까지 기록하고 mixedContentCheck가 0이면서 Ccount 값이 있으면 하위 엘리먼트의 SORD 값을 가지는 엘리먼트로 이동하여 시작 태그를 기록한다. mixedContentCheck가 1이면 Ccount 값은 반드시 존재하므로 내용이 먼저 나오는 경우 내용을 기록하고 하위 엘리먼트로 이동한다. "<"가 먼저 나오면 해당 SORD 값을 갖는 하위 엘리먼트로 이동한다.

(그림 12)는 문서의 일부분을 통합하는 과정으로 SORD 값으로 정렬을 한 후에 CONTENT 부분의 시작 엘리먼트



(그림 12) 문서 병합 예

를 버퍼에 기록하고 mixedContentCheck 값과 Ccount 값을 검사하여 앞에서 설명한 대로 문서를 통합하는 과정을 보여주고 있다.

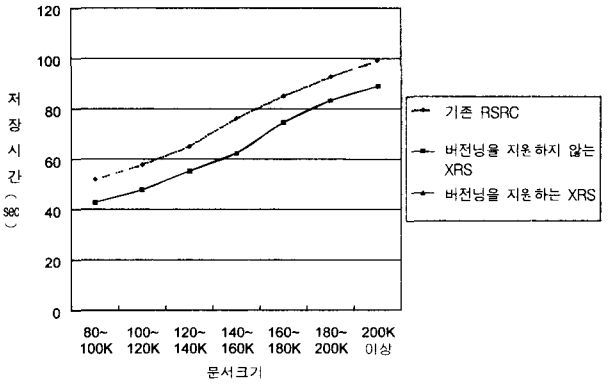
5. 성능 평가

이 절에서는 구현한 버전을 지원하는 XML 저장관리 시스템의 성능평가 결과를 기술한다. 기존 버전기법들은 I/O 오버헤드 관점에서 페이지 접근횟수를 계산하여 성능평가를 실시하였는데, 본 논문에서는 DBMS 기반으로 구현하여 기존 버전기법들과의 성능평가가 의미가 없어 [7]에서 제안한 분할모델을 기반으로 한 XML 저장관리 시스템과 버전을 지원하지 않는 제안한 방법, 버전을 지원하는 제안한 방법에 대해 문서 저장과 문서 검색 시간 측면 및 몇가지 질의에 대해 비교 수행하였다. <표 1>은 본 논문에서 사용된 테스트 문서의 분석 결과를 나타낸다. 구현한 시스템의 성능평가는 학위논문의 형식에 맞게 설계한 paper.dtd를 가지고 이미지를 제외한 총 653M 분량의 문서 2,975개를 가지고 SUN Enterprise 250에서 저장 테스트한 결과이다. 각 문서를 구조에 따라 나누었을 때 총 3,964,565개의 엘리먼트들이 생성되었다. 또한 각 XML 문서당 이미지의 평균 개수는 13개, 평균 크기는 37.5K이고 각 버전들은 20%의 문서변경을 고려하였다. 본 논문에서는 성능평가에서 사용하고 있는 기존 XML 저장관리 시스템을 RSRC(Repository System based on RDBMS and Composition Model)으로 명명하고, 구현한 XML 저장 시스템을 XRS(XML Repository System)라 명명한다.

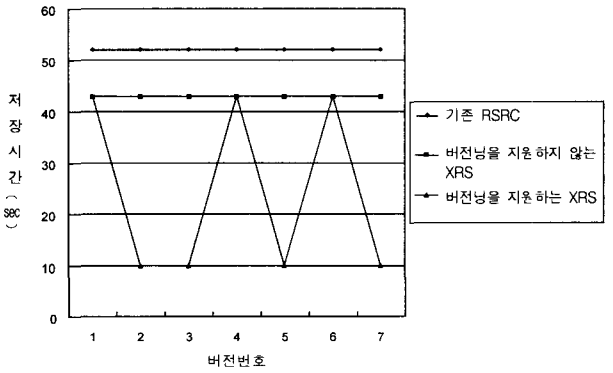
(그림 13)은 문서 크기에 따른 저장 시간을 나타낸다. 기존 RSRC의 평균 삽입시간은 75.34초이나 본 논문에서 구현한 시스템의 평균 삽입시간은 64.05초로 약 12% 정도의 저장 속도가 향상되었다. 이는 하나의 ELEMENT 테이블에

〈표 4〉 테스트 문서의 분석 결과

의 미	값
전체 문서 개수	2,975
평균 문서 크기	37.5K
문서당 평균 엘리먼트 개수	1,332
문서당 평균 애트리뷰트 개수	250
전체 엘리먼트 개수	3,964,565
전체 키워드 개수	15,862,397



(그림 13) 문서 저장 시간 1

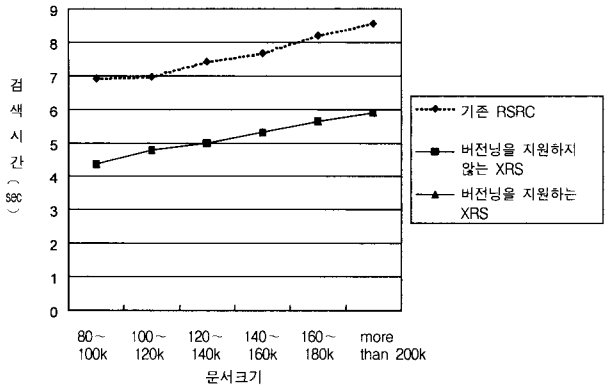


(그림 14) 문서 저장 시간 2

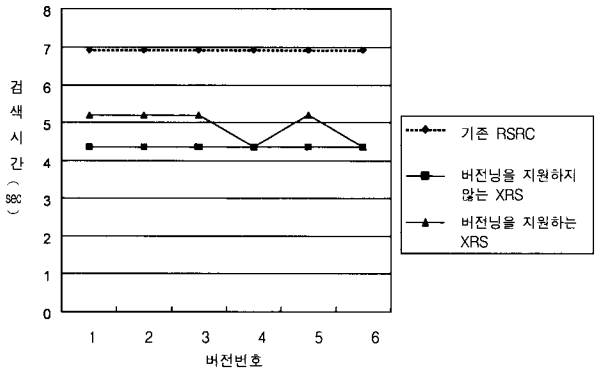
모든 정보를 저장하기 때문이다. (그림 14)는 버전닝을 고려하여 버전횟수에 따라 저장시간을 나타낸다. 버전닝이 발생하는 경우에 변경된 부분만 저장되기 때문에 저장시간이 적게 걸리고, 일부 버전들이 버전닝을 지원하는 않는 XRS와 같은 이유는 XML 문서의 구조가 변경되어 전체적으로 저장되기 때문이다.

(그림 15)는 문서 전체를 추출하는데 있어서 문서 크기에 따른 검색시간을 나타낸 것이다. 기존 RSRC의 경우 해당 문서를 구성하는 모든 엘리먼트, 애트리뷰트 등을 추출하여 그들간의 관계를 구성하면서 엘리먼트 이름을 가지고 직접 시작 엘리먼트와 끝 엘리먼트를 작성하면서 추출하기 때문에 상당히 큰 검색시간을 가진다. 그러나 본 논문에서 제안한 분할 모델은 SORD 값으로 정렬한 후 하위 엘리먼트로 순회하면서 CONTENT 필드에 있는 시작 태그와 PCDATA를 기록하고 상위 엘리먼트를 순회하면서 남아있는 PCDATA

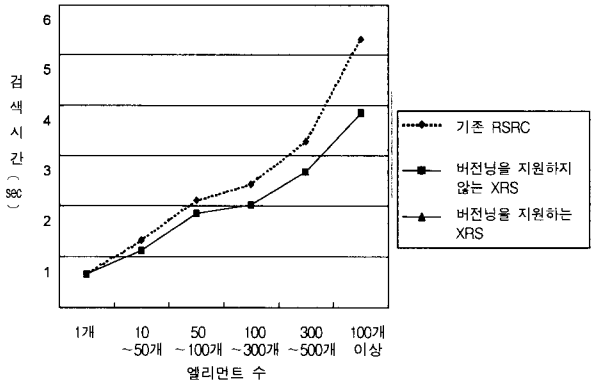
와 끝 태그를 기록함으로써 빠른 추출이 가능하였다. (그림 15)에서 보는바와 같이 성능평가 결과 문서 전체에 대한 추출은 평균 15%의 향상을 가져왔다. 문서 일부분에 대한 추출 결과는 단말노드 추출 시는 같은 검색 결과를 보였으며, 비단말 노드의 추출시는 하위에 포함되는 엘리먼트의 개수에 영향을 받음을 (그림 17)을 통해 알 수 있었다. (그림 16)은 버전닝을 고려한 문서 검색 시간을 보여 준다. 버전닝이 발생하는 경우에 변경된 부분만 저장되어 있기 때문에 변경된 부분에 대한 병합하는데 걸리는 시간만큼 더 걸린다.



(그림 15) 문서 검색 시간 1



(그림 16) 문서 검색 시간 2



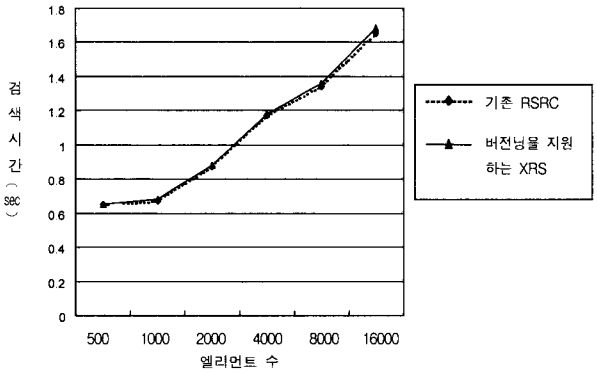
(그림 17) 문서 일부분 검색 시간

다음은 내용 검색과 구조 검색에 대한 질의에 대해 성능 평가를 실시하였다. 사용된 질의는 다음과 같다.

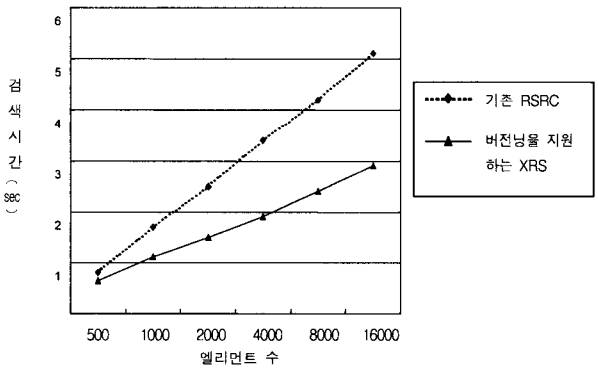
질의 1 : title 엘리먼트가 "XML" 키워드를 포함하는 엘리먼트를 검색하시오.

질의 2 : "구조검색"을 포함하는 section 엘리먼트의 부모 엘리먼트를 검색하시오.

(그림 18)은 질의 1의 검색 결과를 나타내고, (그림 19)는 질의 2의 검색 결과를 나타낸다.



(그림 18) 내용 검색 시간



(그림 19) 구조 검색 시간

내용 검색의 경우, 많은 차이가 나지 않음을 볼 수 있다. 하지만 구조 검색의 경우, 많은 차이가 생기는 것을 알 수 있다. 그 이유는 포함 관계를 통해 중간 결과 집합을 생성하지만 그 결과 중에 정확한 질의의 결과에 해당하는 부모 엘리먼트인지를 검사해야 하기 때문에 검색 시간이 더 걸린다.

6. 결 론

본 논문에서는 문서의 수정을 효율적으로 지원하는 분할 모델을 이용하여 문서 수정에 따른 버전닝을 지원하는 데이터 모델을 제안하고, 버전닝을 지원하는 XML 저장관리 시스템을 설계하고 구현하였다. 지원되는 버전관리는 문서의 구조는 변경되지 않고 단지 내용만이 수정되는 경우와 문서의 구조가 바뀌면서 내용이 수정되는 경우로 나누어진다. 내용

만 변경되는 경우에는 변경된 부분에 대해서만 버전을 유지하는 엘리먼트 단위버전을 제공하고, 구조 변경시에는 구조 정보가 달라지기 때문에 문서 단위 버전을 제공한다.

향후 연구 방향으로는 현재 구조 변경시에는 문서 전체에 대해서 버전을 유지하는데, 구조 변경된 부분에 대한 구조 검색을 지원할 수 있도록 동적으로 구조정보를 표현하는 연구가 필요하다. 또한 버전닝을 고려한 타 XML 저장관리 시스템과의 성능평가를 실시하는 것이다.

참 고 문 헌

[1] Hyung-Il Kang, Jae Soo Yoo, ByoungYup Lee, "Design and Implementation of a XML Repository System Using DBMS and IRS," XML Asia Pacific, 2000.

[2] Walter F. Tichy, "RCS-A SYstem for Version Control," Software-Practice & Experience, 15, 7, pp.637-654, July, 1985.

[3] Shu-Yao Chien,T sotras V. J., Zaniolo. C, "Copy-based versus edit-based version management schemes for structured documents," Research Issues in Data Engineering 2001, Proceedings, pp.95-102, 2001.

[4] Ricardo Baeza-Yates and Gonzalo Navarro, "Integrating Contents and Structure in Text Retrieval," ACM SIGMOD, pp.67-79, 1996.

[5] K. Bohm, A. Muller, and E. Neuhold, "Structured Document Handling - a Case for Integration Databases and Information Retrieval," CIKM '94, pp.147-154, 1994.

[6] V. Christophides, S. Abiteboul, S. Cluet, and M. Schol, "From Structured Documents to Novel Query Facilities," SIGMOD, pp.313-324, 1994.

[7] Daniela Florescu and Donald Kossmann, "Storing and Querying XML Data using an RDBMS," IEEE Data Engineering Bulletin, Vol.22, No.3, 1999.

[8] Tuong Dao, Ron Sacks-Davis, and James A. Thom, "An Indexing Scheme for Structured Documents and its Implementation," Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA '97), pp.125-134, 1997.

[9] R. Goldman, J. McHugh, and J. Widom, "From Semistructured data to XML : Migrating the Lore data model and query language," Proc. of the WebDB Workshop, 1999.

[10] Lee, Y. K., Yoo, S. J., Yoon, K. R and Berra, P. B., "Index Structures for Structured Documents," Proc. Digital Library '96, pp.91-99, 1996.

[11] Sung-Geun Han, Jeong-Han Son, Jae-Woo Chang and Zong-Chel Zhoo, "Design and Implementation of a Structured Information Retrieval System for SGML documents," Database Systems for Advanced Applications, pp.81-88, 1999.

[12] Takeyuki Shimura, Masatoshi Yoshikawa, and Shunsuke zUemura, "Storage and Retrieval of XML Documents Using Object-Relational Databases," DEXA, 1999.

[13] Schoning, H., "Tamino-a DBMS designed for XML," Data Engineering 2001, Proceedings, 17th International Conference, pp.149-154, 2001.

[14] Cheng J., Xu J., "XML and DB2," Data Engineering 2000, Proceedings, 16th International Conference, pp.569-573, 2000.

[15] Oracle, Data Engineering, 16th International Conference, 2000.

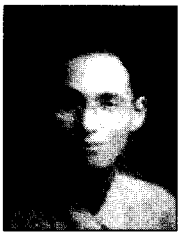
[15] Banerjee S. etc, "Oracle8i-the XML enabled data management system," Data Engineering 2000, Proceedings. 16th International Conference, pp.561-568, 2000.

[16] eXcelon, http://www.exceloncorp.com.

[17] Sung Wan Kim, etc, "Developing a native storage structure for XML repository system in main memory," High Speed Networks and Multimedia Communications 5th IEEE International Conference, pp.96-100, 2002.

손 충 범

e-mail : cbson@trut.chungbuk.ac.kr
1997년 충북대학교 정보통신공학과(공학사)
1999년 충북대학교 정보통신공학과(공학석사)
1999년~2002년 충북대학교 정보통신 공학과(공학박사)
관심분야 : 데이터베이스 시스템, XML, 정보검색, 분산객체 컴퓨팅



오 경 근

e-mail : elaborator@gainit.co.kr
1998년 충북대학교 정보통신공학과(공학사)
1999년~2000년 컴퓨터학원 강사
2000년~2002년 충북대학교 정보통신공학과 (공학석사)
2002년~현재 가인정보기술 연구소 연구원
관심분야 : 데이터베이스 시스템, XML, WML, 분산객체 컴퓨팅, NGOSS, NMS

유 재 수

e-mail : yjs@cbucc.chungbuk.ac.kr
1989년 전북대학교 컴퓨터공학과(공학사)
1991년 한국과학기술원 전산학과(공학석사)
1995년 한국과학기술원 전산학과(공학박사)
1995년~1996년 목포대학교 전산통계학과 전임강사
1996년~현재 충북대학교 정보통신공학과 부교수
관심분야 : 데이터베이스 시스템, XML, 멀티미디어 데이터베이스, 분산객체 컴퓨팅, etc.