XML 문서저장에 관한 민군겸용 데이터베이스 관리체계의 성능비교

강석훈* · 이재윤** · 이말순**

목 차

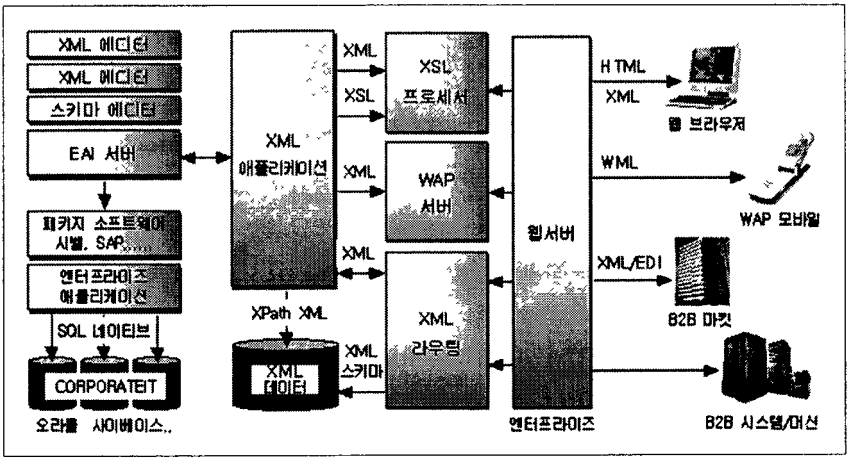
1. 서론
2. 관련 연구
3. XML 문서 저장 방식의 비교
4. 성능평가
5. 결론

1. 서론

다양한 형태의 정보들이 인터넷상에 기하급수적으로 급증하면서 많은 사용자들로부터 기존 정보의 공유와 호환성, 편의성을 효과적으로 사용하고자 하는 연구가 진행되었다. 이러한 요구에 부흥하기 위하여 HTML의 편리한 사용성과 SGML의 확장성 등의 장점을 취

* 대전대학교 컴퓨터공학과 교수
** 대전대학교 컴퓨터공학과 석사

합한 XML (eXtensible Markup Language)이 표준 마크업 언어로 채택되었고 시스템 통합이라는 전기를 맞아 급속한 발전을 이루게 되었다([1],[2]).



<그림 1> 민간기업과 군 컴퓨팅 환경에서의 XML 사용

민간기업과 군에서도 그림 1에 도시한 바와 같이 XML 문서들이 많아짐에 따라 XML 문서들을 통합적인 관리 또한 필요하게 되었고 DBMS 들도 이러한 추세에 맞추어 한 단계 발전하게 되었다[3].

따라서 군에서도 급속히 발전하고 있는 웹 또는 컴퓨터상의 서로 다른 종류의 다양하고 많은 양의 정보를 제공하기 위해서는 문서들을 전산화하는 것과 다량의 전자 문서를 저장 관리하고 검색을 할 수 있는 시스템이 필요하다. 이러한 문서 저장 관리기를 통해 대량의 문서를 저장하고 저장된 문서 중 사용자가 원하는 부분에 대해 빠른 검색을 지원하며, 저장된 문서의 일부분을 추가, 삭제 및 변경할 수 있어야 한다[4].

기존의 DBMS는 문서의 내용정보에만 중점을 두어 문서를 저장 관리하였다. 그러나 최근의 XML과 같은 구조문서는 문서의 내용뿐만 아니라 그러한 내용이 무엇과 관련되어 있는지에 대한 구조정보도 문서에 포함하고 있기 때문에, 이러한 문서를 처리해줄 수 있는 DBMS를 필요로 하게 되었다 XML 문서 저장이 가능한 백엔드 데이터 베이스 시스템으로는 RDBMS와 OODBMS 그리고ORDBMS 등이 있지만 이러한 시스템들은 엘리먼트를 테이블로 표현 시 많은 조인들을 유발하므로 성능 저하를 유발하는 문제점을 갖는다.

이러한 성능 저하 문제를 해결하기 위하여 Native XML DBMS에서는 XML 문서를 테이블로 표시하지 않고 XML 형태를 유지하여 저장함으로서 기존 시스템들에서의 성능 저하 문제를 개선하는 접근 방법을 채택하였다.

Native XML DBMS를 이용하여 데이터를 저장할 경우, 계층적인 조인을 이용한 것보다 빠르게 처리할 수 있고, XML 문서를 Native하게 저장함으로서 대용량의 데이터에 대한 질의를 안정성 있게 처리할 수 있다는 장점을 가진다.

그러나 관계형 DBMS는 현재 사용자들의 과반수 이상을 차지하는 제품으로서, 새로운 XML DBMS로의 이전을 하기 위해서는 추가비용이 요구되는 부담이 있다. 그러므로 본 논문에서는 가장 많이 구현된 방식으로서 XML의 계층 구조를 관계형 데이터베이스의 관계로 표현하며 XML의 각 요소와 속성을 테이블의 필드로 표현하는 관계형 데이터베이스 맵핑형과 XML 문서 처리의 강력한 기능을 가지는 새로운 저장 기법으로 등장한 Native XML DBMS를 비교 평가한다[5].

본 논문의 구성은 다음과 같다. 2절에서는 연구와 관련된 성능평가 모델의 XML 문서저장 방식들과 해당 DBMS에 대하여 살펴보고, 3절에서는 RDBMS와 OODBMS 그리고, Native XML DBMS

의 저장 방식에 대해서 기술하고, 4절에서는 DBMS 유형별로 성능을 비교 평가한다. 마지막으로 5절에서 결론 및 향후 연구방향을 제시한다.

2. 관련 연구

과거 저장 모델과 저장 시스템에 대한 연구가 활발히 진행되어 왔으며, 최근 XML은 차세대 저장 모델로 대두되어 왔다. 이번 절에서는 기존 XML저장 방법에 대하여 알아보고, 민군겸용이 가능한 DBMS에서 XML 문서저장 방식을 비교한다.

2.1 XML 지원 DBMS의 분류

2.1.1 관계형 데이터베이스 시스템

관계형 데이터베이스는 일련의 정형화된 테이블로 구성된 데이터 항목들의 집합체로서, 그 데이터들은 데이터베이스 테이블을 재구성하지 않더라도 다양한 방법으로 접근하거나 조합될 수 있다.

이러한 관계형 데이터베이스 시스템을 이용한 방법은 XML 문서의 저장을 위한 구조정보를 관계데이터베이스에 테이블 형태로 구성하여 저장한다. 이를 이용한 방식으로는 첫째로 LIST형태를 이용한 방법, 둘째로 경로 엘리먼트 ID를 이용한 방법, 셋째로 DFS Numbering을 이용한 방법, 넷째로 UID를 이용한 방법 등이 있다. 그런데 관계 데이터베이스를 이용하는 방법은 RDBMS의 우수한 성능을 이용할 수 있고, 기존의 응용시스템의 데이터를 함께 사용할 수 있는 장점을 가진다. 그러나 검색 시 다수의 테이블에 대한 고비용의 조인 연산을 수행하여야 하며 set-value를 지원하지 않는

등의 단점을 가진다.

2.1.2 객체 지향형 데이터베이스 시스템

객체지향 데이터베이스는 XML 데이터의 계층 구조를 객체지향의 클래스계층에 매핑하는 방법으로 XML 데이터를 XML 파서를 통하여 DOM으로 객체화하여 데이터베이스에 저장, 관리하는 것으로 XML 문서를 하나의 데이터로 관리하는 방법이다. XML의 데이터 모델이 객체지향 데이터 모델과 유사하므로 거의 모든 XML 데이터 구조를 저장 할 수 있다.

객체지향 방법은 관계형 데이터베이스를 사용하는 방법에 비해 XML문서에 대한 모델링에 적합하며, 엘리먼트들 간의 전후종속 관계를 클래스 구조를 이용하여 쉽게 적용할 수 있다.

엘리먼트에 대한 접근시 관계형 데이터베이스를 이용할 경우 DFS를 이용한 UID 계승 엘리먼트 ID값을 키로 하여 접근이 가능하지만, 반면에 객체지향 데이터베이스의 경우 이를 위해 특별한 모델을 제공하지 않을 경우 데이터베이스내의 모든 객체를 탐색하게 된느 경우가 발생하게 된다 이러한 문제점을 극복하기 위해 데이터베이스와 파일 시스템을 연동하여 사용하는 경우 검색효율은 증가하나 두 개의 시스템을 동시에 개발하여야 하며, 두 시스템내의 일관성을 유지하기가 어렵게 된다.

2.1.3 Native XML 데이터베이스 시스템

"네이티브 XML 데이터베이스(Native XML database)"란 단어는 Software AG사의 Tamino XML 데이터베이스 제품 캠페인에서 처음으로 나온 말이다. 그리고, Software AG사의 Tamino XML 데이

터베이스 제품 캠페인의 성공 덕분에 "네이티브 XML 데이터베이스(Native XML database)"란 단어가 같은 제품을 개발하는 업체들 사이에 공통적인 용어로서 자리 잡게 된 것이다. 네이티브 XML 데이터베이스라는 용어가 마케팅 차원의 캠페인에서 나오다 보니, 공식적으로 기술적인 정의를 가지지 못했다. 그러나, 네이티브 XML 데이터베이스가 업계의 표준 용어로서 자리 잡게 되면서, XML .DB[10] 메일링 리스트의 멤버들에 의해 네이티브 XML 데이터베이스에 대한 정의가 만들어졌고, 이 정의를 살펴보면, 다음과 같다.

1. XML 도큐먼트(도큐먼트 내의 데이터와는 대조적으로)의 논리적인 모델을 정의하고 그 모델에 따라 도큐먼트를 저장하고 가져온다. 모델은 최소한 반드시 엘리먼트, 에트리뷰트, PCDATA, 그리고 도큐먼트 순서 등을 포함하고 있어야 한다. 이러한 모델의 예를 들면, XPath 데이터 모델, XML 인포셋(XML Infoset), 그리고 DOM과 SAX 1.0 이벤트 등에 의해 넌지시 암시된 모델 등이 있다.
2 논리적인 스토리지의 가장 기본적인 단위로서 XML 도큐먼트를 갖는다. 반면, 관계형 데이터베이스에서는 논리적인 스토리지의 가장 기본적인 단위로서 테이블 내의 행(row)을 갖는다
3. 어떤 특정한 하부의 물리적인 스토리지 모델을 가질 필요는 없다 예를 들어, 관계형, 계층형, 또는 객체-지향 데이터베이스 상에 구축될 수도 있고, 또는 인덱스화(indexed), 압축된(compressed) 파일 등과 같은 독점적인(proprietary) 스토리지 포맷을 사용할 수도 있다

첫 번째 정의 부분은 데이터베이스에 의해 사용된 모델에 관심을 갖는다는 점에서 다른 타입의 데이터베이스 정의와 유사하다. 주어

진 네이티브 XML 데이터베이스가 사용하는 모델 내에 포함되어 있는 것보다 더 많은 정보를 저장할 수 있을 것이다. 예를 들어, XPath 데이터 모델에 기반한 쿼리를 지원하지만, 텍스트로 도큐먼트를 저장한다. 이런 경우, CDATA 섹션 및 엔티티 유사지와 같은 것들은 데이터베이스 내에 저장되지만, 모델에는 포함되지 않는다.

두 번째 정의는 네이티브 XML 데이터베이스 내에 있는 기본적인 스토리지 단위가 XML 도큐먼트라는 것을 나타낸다. 네이티브 XML 데이터베이스가 도큐먼트 조각(document fragments)에 이러한 역할을 할당할 수 있고, 오늘날 모든 네이티브 XML 데이터베이스 내에는 도큐먼트에 의해 채워진다 스토리지의 기본적인 단위는 주어진 데이터 조각이 채워질 최하위 레벨의 컨텍스트라 할 수 있고, 이는 관계형 데이터베이스에서의 행(row)과 동등하다 할 수 있다. 이는 도큐먼트 프레그먼트, 개개의 엘리먼트 등과 같은 보다 작은 단위의 데이터를 검색하는 것을 가능하게 하고, 또한 다른 도큐먼트를 위한 프레그먼트를 갖는 하나의 도큐먼트로부터 프레그먼트들을 결합하는 것을 가능하게 한다. 관계형 데이터베이스에서 행이 존재함으로써 개개의 컬럼 값을 가져오거나 또는 기존의 행들 사이에 새로운 행을 생성하는 것이 가능한 것과 같다고 할 수 있다.

세 번째 정의는 하부의 데이터 스토리지 포맷은 중요하지 않다는 것을 나타낸다 다시 말해서, 관계형 데이터베이스에 의해 사용된 물리적인 스토리지 포맷이 데이터베이스가 관계형인가 여부는 아무런 상관이 없다는 것을 의미한다.

2.2 민군겸용 DBMS의 XML 문서 처리 방식

현재 대부분의 DBMS 회사들은 XML 문서를 수용할 수 있게 확장하고 있다. ObjectStore를 확장해서 만든 eXcelon[2]은 XML 문

서를 개별적인 XML 엘리먼트로써 객체들로 저장한다. 오라클사의 오라클 9i 서버[3]와 함께 사용되는 오라클 XML DB는 XML 데이터를 컬럼의 데이터타입으로 저장 할 수 있도록 XML Type을 지원하고, XML 데이터 및 문서의 관리를 위해 XML Repository라는 인터넷 저장소를 제공한다. 또한 XML 전용 DBMS인 AG의 Tamino[4]는 컬렉션 형태로 여러 개의 문서 형태로 구성되어 관리되어진다.

기존의 데이터베이스 기반의 XML 문서 저장 시스템 연구는 크게 XML 문서를 관계 또는 객체지향 데이터 모델로 변환하여 저장하는 모델링 연구와 효율적인 검색을 위한 인덱스 및 검색 구조 설계 연구로 나뉜다. 기존의 시스템은 XML 문서를 데이터 모델로 변환하는 과정에서 XML의 구조적 특징과 내용을 문서 독립적으로 모델링하여 저장하지 않으므로 XML 문서 구조의 갱신이 발생할 경우, 이에 따른 모든 구조정보가 수정되어야 했고, DTD. 정보를 저장함에 있어 DTD 트리의 순차적인 탐색을 기반으로 리스트 형태의 정보를 데이터베이스에 저장하므로, 검색시의 효율이 떨어지거나 임의 위치의 엘리먼트 탐색이 어렵다.

이에 비해 Native XML 저장방법을 이용하는 경우 XML 엘리먼트를 맵핑하는 과정이 필요 없이 그대로 저장하므로 저장 시 부하가 최소화 되고, 조회 시 계층 구조에 최적화 되어 있고 XML 데이터를 있는 그대로 가져오므로 조회 시 부하가 경감되며 XML 구조 변경 시 데이터베이스 스키마 변경이 쉬운 장점을 가지는 반면 Native XML DBMS에 저장 시 질의 결과가 XML 문서라는 면과 관계형 DB에서 ODBC나 JDBC와 같은 표준화된 API가 개발되지 않았다는 개발 환경상의 단점을 가진다[6].

3. XML 문서 저장 방식의 비교

RDBMS와 Native XML DBMS에서의 XML 문서의 저장 방식을 비교하고 두 저장 방식의 특징을 찾아내어 XML 문서에 따른 적합한 저장 방식을 제시한다.

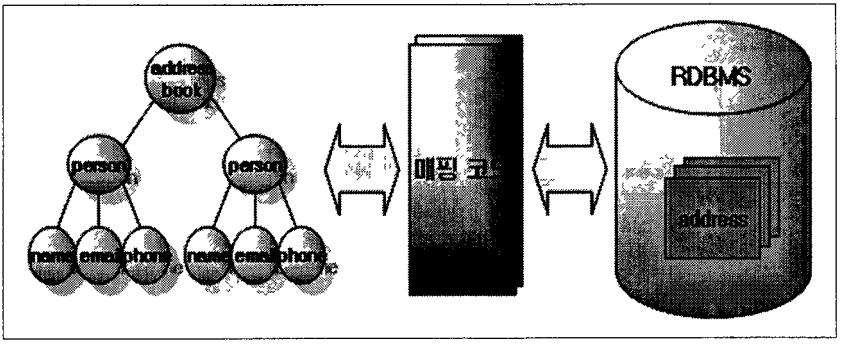
3.1 RDBMS에서의 XML문서 저장방식

그림 3와 같이 RDBMS에서 XML 문서를 저장하기 위해서는 매핑단계를 거친다. 매핑을 위해서는 미들웨어를 사용하여 매핑을 하거나 XML 가능 데이터베이스와 미들웨어를 사용하여 저장한다. 매핑하는 방법은 Table-Based Mapping과 Object-Relation Mapping으로 분류된다. 테이블 기반 매핑 방법은 소프트웨어를 사용하여 column에 child elements와 attributes 데이터로 저장되고 각 element나 attribute는 XML 문서에서의 태그들의 이름으로 사용된다. 테이블 기반 매핑은 연속적인 관계 데이터를 나타내기에 유용하지만 형식에 맞지 않는 XML 문서에는 사용할 수 없는 단점을 가진다. 객체 관계 매핑은 XML 문서를 객체 트리처럼 데이터를 저장하는 모델로 엘리먼트와 애트리뷰트 type, element content, complex element type이 클래스화 되어 테이블에 매핑된다[7].

3.2 객체지향 DBMS

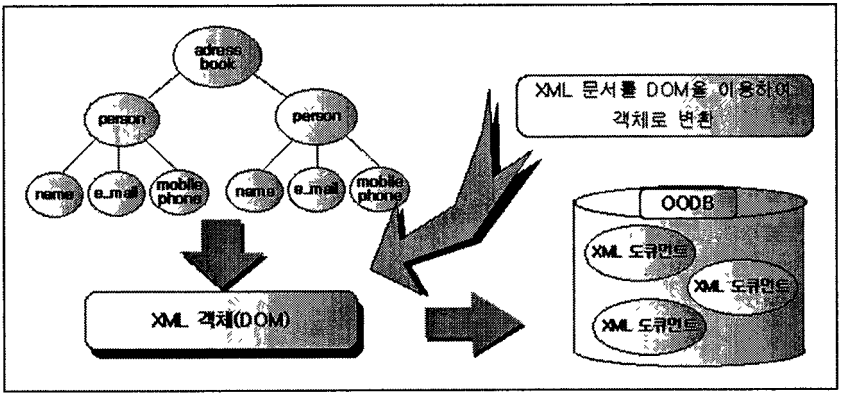
객체지향 DBMS는 그림 4에서 볼 수 있듯이 XML 데이터의 계층구조를 객체지향의 클래스 계층에 매핑하는 방식이다. XML의 데이터 모델이 객체지향의 데이터 모델과 유사하므로 거의 모든

XML 데이터 구조를 저장할 수 있다.



<그림 3> RDBMS에서의 XML문서 저장방식

즉, 디자인 단계에서의 자료구조와 응용 프로그램을 작성할때의 자료구조, 및 DBMS에 저장된 자료구조의 형태가 동일하기 때문에 각 단계간에 변환이 필요하지 않다 그 외에도 디자인 단계에서의 모델링에 다양한 개념들을 지원하고, 메소드의 수행과 같은 여러 기능들이 있다.

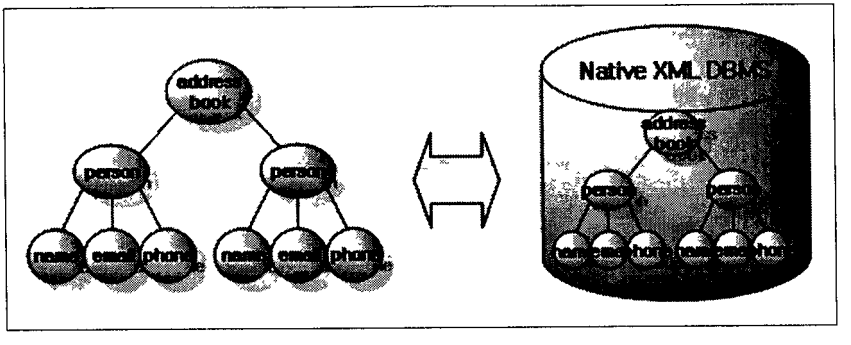


<그림 4> OODBMS에서의 문서 저장 방식

<그림 4>의 객체지향 DBMS에서의 XML 문서 저장 방식을 살펴보면, XML 데이터를 XML 파서로 DOM 객체화 하고 데이터베이스에 저장, 관리하는 것으로 XML 문서를 하나의 데이터로 관리한다([8],[9]).

3.3 Native XML에서의 문서 저장방식

Native XML 데이터베이스에서는 XML문서를 기본단위로 가지는 스토리지(storage)라는 저장 개념을 이용하며 두가지 아키텍쳐로 분류한다. 먼저 텍스트 기반 스토리지는 전체 XML 도큐먼트를 텍스트 형태로 저장하고, 이는 파일시스템 내의 파일 일수도 있고 RDBMS의 BLOB 또는 독점적인 텍스트 포맷일수도 있다. 도큐먼트를 액세스 할 수 있는 몇 가지 종류의 데이터베이스 기능을 제공한다. 두번째 모델 기반 스토리지는 XML 도큐먼트(docume nt)를 텍스트로 저장하기 보다는 도큐먼트로부터 내부 객체를 생성하고 이 모델을 저장한다 즉, DOM 또는 DOM의 변종과 같은 도큐먼트의 바이너리 모델을 기존의 데이터 저장소에 저장하는 것이다([10],[11],[12])



<그림 5> Native XML에서의 문서 저장 방식

<표 1> Native XML DBMS와 RDBMS의 비교

구분	RDBMS
저장	-XML 태그를 각 테이블의 컬럼으로 맵핑 -RDBMS 테이블의 특성상 정규화 과정 필요 -XML 문서의 계층구조가 깊어질수록 테이블 구조가 복잡해짐 -XML의 속성을 표현할 방법이 없음
조회	-조회시에는 쪼개진 테이블을 join해서 가져 와야 하기 때문에 부하 발생 -조회하는 SQL 문장이 복잡함
관리	-XML 구조 변경시 데이터베이스 스키마 변경이 어려움
구분	OODBMS
저장	- XML 문서 계층 구조를 클래스 계층으로 맵핑 - XML의 데이터 모델과 객체지향의 데이터 모델이 유사
조회	- 레코드와 레코드 사이 데이터 검색이 포인터(Pointer)에 의하여 이루어지므로 성능 문제가 대두되지 않음 - XPath 쿼리 사용 - 인덱싱 기능 사용으로 쿼리 성능 향상
관리	- XML 구조 변경시 데이터베이스 스키마 변경이 쉬움
구분	Native XML DBMS
저장	-XML 문서 맵핑 없이 자체를 그대로 저장 -XML 데이터를 있는 그대로 저장하므로 저장시 부하가 없음 -XML 객체나 요소를 원래의 형태대로 저장
조회	-조회시 계층(hierarchy)구조에 최적화 됨 -XQL,XPath,X_Query등 XML을 위한 쿼리 사용 -조회하는 쿼리가 간단함 -XML데이터를 있는 그대로 가져오므로 조회시 부하가 없음
관리	-XML 구조 변경시 데이터베이스 스키마 변경이 쉬움

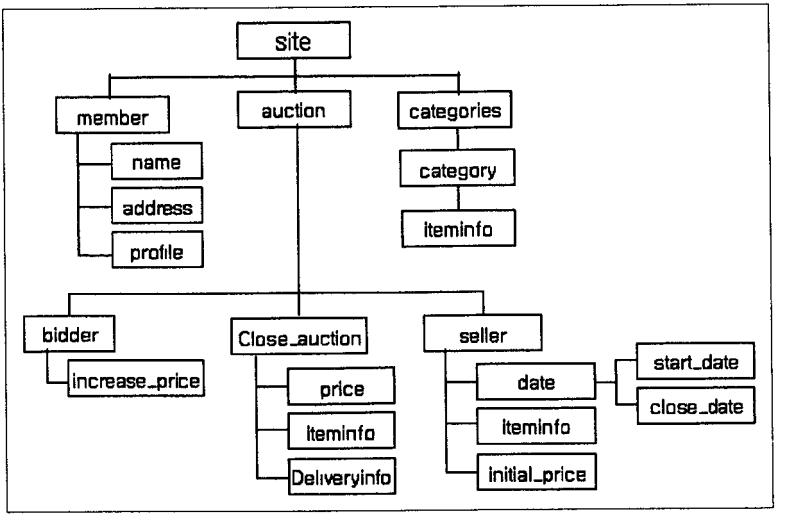
4. 성능평가

DTD 독립적인 문서와 의존적인 XML 문서의 평가를 위해 DTD를 설계하고, 데이터 중심 문서와 도큐먼트 중심의 예제를 들어

Native DBMS와 관계형 DBMS에 각각 저장, 검색하고 수행시간을 측정하였다.

```
<!ELEMENT Shopping_mall (#PCDATA)>
<!ELEMENT SHOPPING (member, auction, categories, ……)>
<!ELEMENT member (Shopping_mall)>
<!ELEMENT auction (Shopping_mall)>
<!ELEMENT categories (Shopping_mall)>
                                    .
                                    :
```

<그림 6> 쇼핑몰 DTD



<그림 7> 쇼핑몰 Site 구조

4.1 성능평가 환경

본 실험에서는 비교 평가를 위해 특정 크기의 XML 문서를 저장

하고, 검색하는 수행시간을 측정하였으며, Window 2000 Pentium 700MHz의 환경에서 JAVA를 근거하여 구현되었다. 저장 하부 시스템으로는 Oracle 8i(관계형 DBMS)와 eXcelon(객체지향 DBMS) 그리고, Tamino(Native XML DBMS)를 이용하여 비교 평가 하였으며, 실험에 사용된 XML 문서는 xmlgen을 사용하여 생성하였다.

4.2 성능평가 기준

성능 평가는 앞서 제시한 XML 문서 저장 시스템하에서 조건을 달리하여 성능차이를 비교하였다. 본 시스템에서는 DTD 독립적인 XML 문서와 의존적인 문서간 저장시간과 검색시간의 차이를 비교하고, 검색을 위해 질의의 복잡도에 따른 몇 가지의 질의를 제시하고, 저장시스템의 유형별 관계형 DBMS와 객체지향 DBMS, Native XML DBMS로 저장, 검색시의 차이를 비교하였다

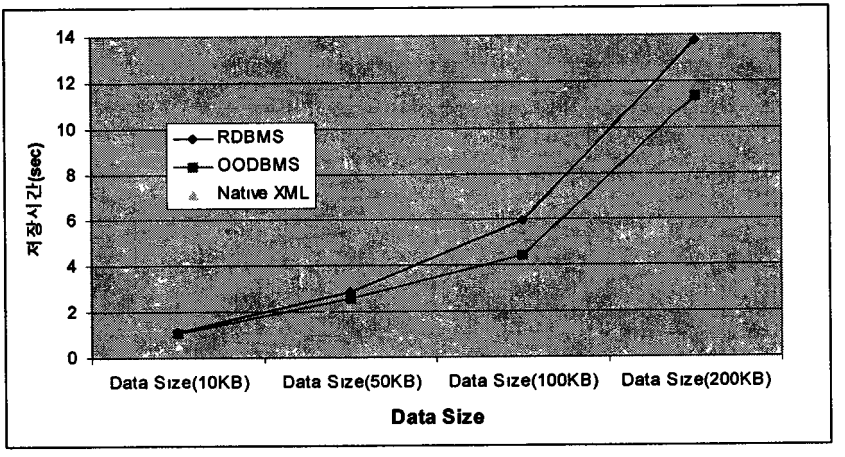
4.3 성능모의 실험

4.3.1 하부 시스템에 다른 데이터 용량별 저장시간 비교

<표 2> 에서는 문서의 크기를 측량값으로 넣어서 DBMS별 저장 결과를 나타낸다.

<표 2> 하부시스템에 따른 데이터 용량별 저장 시간 비교

Size / System	10KB	50KB	100KB	200KB
RDBMS	1 12s	2 84s	5 97s	13 74s
OODBMS	1 17s	2 57s	4.43s	11 39s
Native XML	0 59s	2 2 1s	3 52s	9 86s



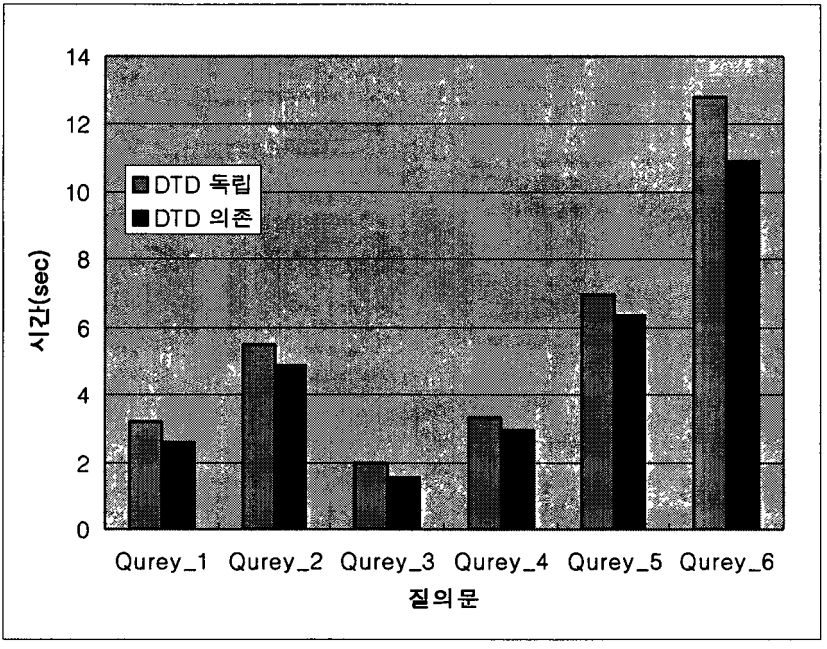
<그림 8> Data Size와 Blockload time

4.3.2 DTD 유무에 따른 검색시간 비교

<표 3>에서는 몇 가지 질의를 통해 DTD의 유무에 따른 처리시간을 나타낸다.

<표 3> DTD 유무에 따른 검색 시간 비교

DTD유무 / Query	DTD 독립	DTD 의존
Query_1	3 18s	2 58s
Query_2	5.48s	4 89s
Query_3	1 99s	1 55s
Query_4	3 31s	2.99s
Query_5	6 99s	6.37s
Query_6	12 80s	10 91s



<그림 9> Data Size와 Blockload time

4.3.3. DTD 유무와 DBMS 유형별 검색시간 비교

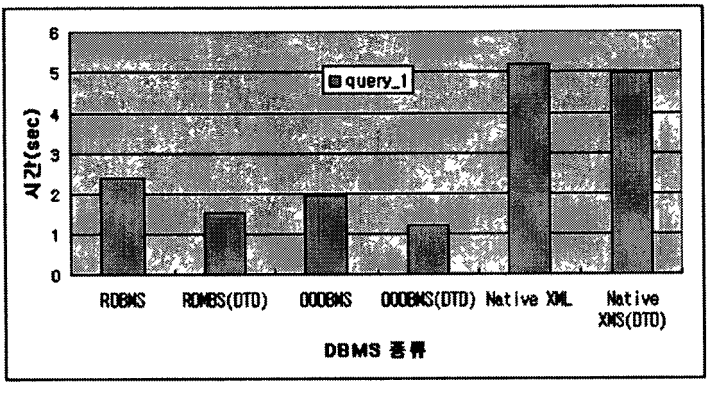
4.3.2 절에서 검색시 사용한 질의에 대한 자세한 내용은 다음과 같다 질의는 단순히 선택 조건을 askwhr하는 검색이나 하나의 XML 문서에서 필요한 부분만을 골라내는 등과 같은 단순질의와 집합연산이나 조인 연산과 같은 2개 이상의 XML 문서를 조합하여 결과를 검색할 수 있는 복합 질의등 질의의 종류를 다양하게 하여 수행하였다.

<표 4> DTD의 유무와 DBMS 유형별 검색 시간 비교

System / Query	RDBMS	RDBMS (DTD)	OODBMS	OODBMS (DTD)	Native XML	Native XML (DTD)
Query_1	2.41s	1.54s	1.97s	1 22s	5.17s	4.98s
Query_2	4.63s	3.79s	4.23s	4.12s	7.59s	6.72s
Query_3	2.21s	2.03s	1.83s	1 29s	1.93s	1.34s
Query_4	3.31s	2.85s	3.27s	3.17s	3.36s	2.94s
Query_5	8.24s	7.03s	7.35s	6.23s	5.37s	3.46s
Query_6	15.89s	12.37s	12.17s	11.64s	10 34s	8 73s

- Query_1 : 단순한 룩업 수행으로 완전히 명시된 경로를 지니는 단순한 문자열을 찾는 질의

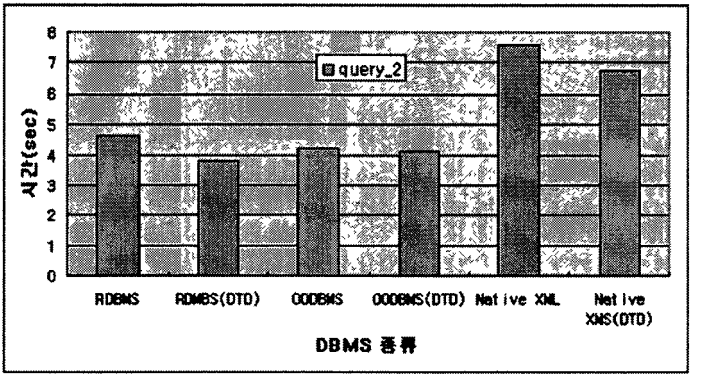
```
FOR $b IN
document("cart.xml")/women/shop/[@id="person0]
RETURN $b/name/text()
```



(a) Query_1 질의 수행 결과

- Query_2 : XML 문서 내부의 순서를 이용해서 접근하는 질의

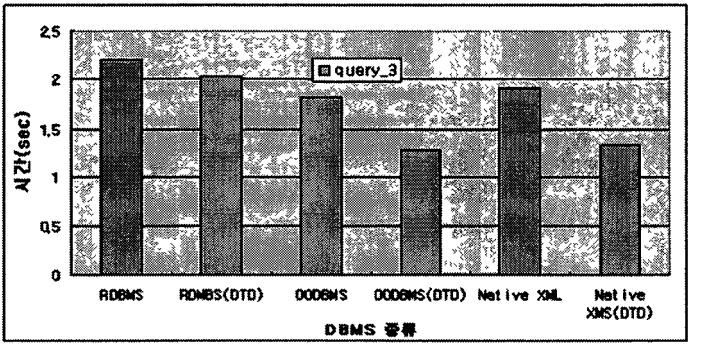
```
FOR  $b IN document("cart.xml")/women/shop
RETURN <increase>
$b/bidder[1]/increase/text()
</increase>
```



(b) Query_2 질의 수행 결과

- Query_3 . 형변환을 포함하는 질의문

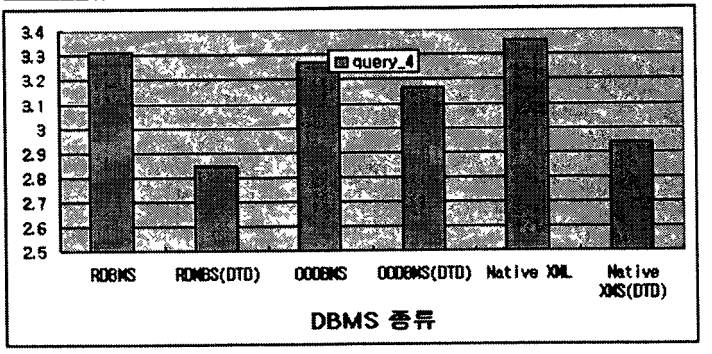
```
COUNT
(FOR $i IN document("cart.xml")/women/shop
WHERE  $i/price/text() >= 40
RETURN $i/price)
```



(c) Query_3 질의 수행 결과

- Query_4 . 정규경로 표현들의 비용을 측정

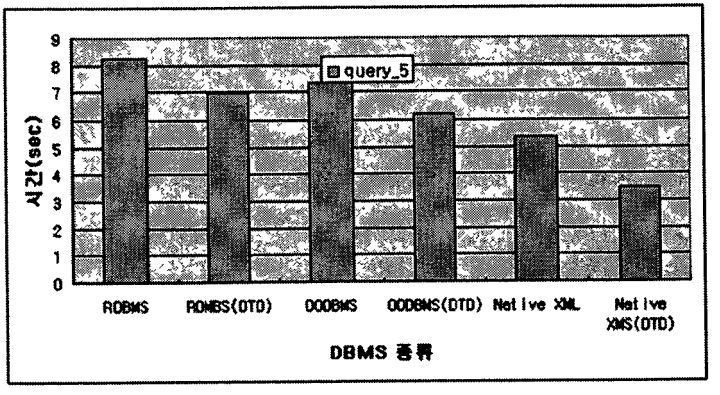
```
FOR  $b IN document("cart.xml")/women/shop
RETURN COUNT ($b//item)
```



(d) Query_4 질의 수행 결과

- Query_5 : 참조추적

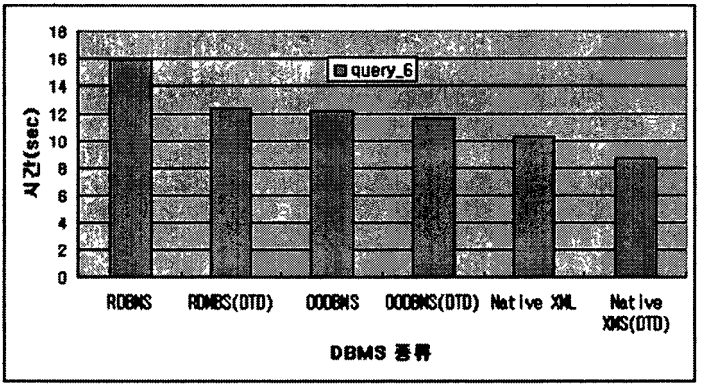
```
FOR $p IN document("cart.xml")/women/shop
LET $a :=
FOR $t IN document("cart xml")/women/shop
WHERE $t/buyer/@person = $p/@id
RETURN $t
RETURN <item person=$p/name/text()>
COUNT ($a) </item>
```



(e) Query_5 질의 수행 결과

- Query_6 : equi-join을 이용한 참조추적

```
FOR     $p IN document("cart.xml")/women/shop
LET     $a :=
FOR $t IN document("cart.xml")/women/shop
        LET $n := FOR $t2
        IN document("cart xml")/women/shop
        WHERE  $t/itemref/@item = $t2/@id
        RETURN $t2
        WHERE $p/@id = $t/buyer/@person
        RETURN <item> $n/name/text() </item>
RETURN
    <person name=$p/name/text()> $a </person>
```



(f) Query_6 질의 수행 결과

<그림 10> DTD의 유무와 DBMS 유형별 검색 시간 비교

4.4 분석평가

4.3절에서는 DBMS의 유형별 성능을 비교하기 위해 Data Size와 DTD의 유무에 따른 저장과 6가지의 다양한 질의를 통한 XML 문서의 검색 시간을 측정하였다. 그 결과 XML 문서 저장에서는 단연 Native XML DBMS의 성능이 우월함을 확인 할 수 있었다. 그러

나, 질의 검색 성능면에서는 차이가 있음을 나타내고 있다. 복잡한 조인 연산이나 검색 Data의 Size가 큰 경우에는 Native XML DBMS의 검색 성능이 약 30% 정도 빠른 검색을 수행하나, 단순 검색이나 단순 집단함수 같은 연산에서는 오히려 관계형 DBMS나 객체지향 DBMS가 더 좋은 성능을 보였다. DTD의 유무에 있어서도 검색의 성능에 큰 차이를 주고 있음을 알 수 있었다. DTD에 의존적인 XML 문서들이 DBMS 유형에 상관없이 DTD 독립적인 XML문서들보다 저장, 검색에서 더 좋은 성능을 가져오는 것을 확인할 수 있었다

5. 결론

본 논문에서는 차세대 저장 방식으로 각광받고 있는 XML 문서를 저장하기 위한 접근 방법들을 민군겸용의 응용을 염두에 두고 분석하고 비교 평가하였다 성능평가의 결과 XML 문서 관리를 위해 데이터베이스의 선택은 XML로 다루고자 하는 데이터의 형식과 데이터의 활용도에 따라 선택되어 져야 함을 알 수 있다

아무리 XML 기반의 기술과 애플리케이션이 주류를 형성하는 추세이기는 하나 기존에 관계형 데이터베이스를 이용하여 데이터를 관리하였다면 반드시 Native XML DBMS를 선택하는 것만이 큰 이점을 가져오는 것은 아니다. 그러나 복잡한 설계 도면과 같은 정보를 관계형 데이터베이스로 계속 관리하는 것은 비효율적일 수 있다. 따라서 XML 데이터를 대량으로 관리하고 이용하는 경우에는 Native DBMS를 권장하고, 기존시스템을 EDI 등에서 XML로 이전하는 경우에는 관계형 데이터베이스를 사용해도 무방하듯이 민군겸용 응용분야별로 적절한 데이터베이스를 선택하여 데이터를 저장하고 관리하는 것이 효과적으로 판단된다.

참고문헌

김채미 · 김재훈. 1998. "XML 기술 및 활용분야". 한국정보처리 학회지 제 5권 제 6호. pp. 66-73.

박용우. 2002. "차세대 e-비지니스 혁명을 이끌 XML 데이터베이스". 프로그램 세계. pp. 272-279.

정희경. 1999. "차세대 웹 문서 표준 XML". 한국정보처리학회 학회지. 제 6권 제 3호 pp. 25-35.

펜타시스템의 Tamino XML Server
http.//www.tamimo.co.kr/taminoserver/jsp/index.jsp

D. Florescu and D. Kossmann, "Storing and Querying XML Data using an RDBMS", IEEE Data Engineering Bulletin 22(3), p.27-34, 1999.

Extensible Markup Language(XML) 1.0 W3C Recommendation, http://www.w3.org/TR/REC-xml.html

eXcelon corporation,
http://www.exceloncorp.com/products/index.htm

Harald Schoning Tamino - A DBMS Designed for XML. In Proc. ICDE Conf. pp. 149-154, 2001.

M. Klettke, H. Meyer, "XML and Object-Relational Database Systems-Enhancing Structural Mappings Based on Statistics", Proc. Web DB Workshop, pp.151-170, 2000.

S. Malaika, "Using XML in Relational Database Applications", 15th Intl conf. on Data Engineering, Sydney, Australia, p 167, 1999.

The Timber Team. The University of Michigan. Available at, http://www.eecs.umich.edu/db/timber

XML and Databases by Ronald Bourret, http://www.rpbourret.com/xml/XMLAndDatabases.htm

Performance Comparison of Database Management Methods on XML Document Storage Functions for both Commerce and Military Applications

Suk-Hoon Kang, Jae-Yun Lee, Mal-Soon Lee

As the research work about XML based on the development of Internet and according to the information exchange standard is being carried out, the need of discovering new methods to store XML documents and manage them efficiently according to the frequency of large-capacity XML documents increases. Consequently, as a kind of back-end database system, XML storage systems such as RDBMS, OODBMS and Native XML DBMS etc. are coming forth in order to save XML documents.

It is an urgent task to make comparisons among usage expense, function comparison storage, inquiry, and manage dimension for each DBMS. This paper makes an analysis and comparison of DTD-independent XML document access methods in RDBMS, OODBMS and Native XML DBMS for XML storage and management. After analyzing the advantages and disadvantages of each access method and comparing the function of typical commerce DBMS such as Oracle 8i, eXcelon and Tamino for finding the possibility of military applications, an another appropriate method to save XML documents is proposed as to find an implementation approach to save structural XML documents.

Key words : XML Database, Performance Analysis.