관계형 데이터베이스와 XML 데이터베이스의 XML 문서 저장 방식에 대한 비교 분석

오정진*, 홍성태**
*전남과학대학 인터넷비즈니스과
**성화대학 디지털컴퓨터계열
Tel : 019-617-3831 / 061-360-5206
E-Mail : jjoh@mail.chunnam-c.ac.kr

A Comparative Analysis on the Storage Method of XML Documents for Relational Database and exclusive XML Database

Jeong-Jin Oh*, Seong-Tae Hong**
*Dept. of Internet Business, Chunnam Techno College
**Dept. of Digital Computer, Sunghwa College
E-Mail:jjoh@mail.chunnam-c.ac.kr/ojj1209@hanmail.net

요약

컴퓨터의 발전과 더불어 자료의 표현 및 저장 방법도 매우 다양해지고 있다. XML은 인터넷의 확산으로 많은 분야에서 연구되어지고 있는 마크업 언어가 되었다. 이처럼 다양한 분야에서 활용되는 XML을 데이터를 관리하는 데이터베이스에 사용되는 것은 당연할 것이다. XML 문서를 데이터베이스에 저장하는 방법에는 파일 시스템으로 저장하는 방법, 관계형 데이터베이스에 저장하는 방법, 개체지향 데이터베이스에 저장하는 방법, XML 전용 데이터베이스에 저장하는 방법 등이 있으며 이들에 대한 연구가 계속되고 있다. 본 논문에서는 XML 문서를 XML이 아닌 다른 형태로 변환하여 저장하는 방법으로 가장 많이 활용되는 관계형 데이터베이스와 XML 문서 형태를 그대로 저장하는 XML 전용 데이터베이스를 비교 분석하는 연구를 하였다.

1. 서론

XML은 인터넷상에서 데이터를 표현하고 교환하는 새로운 표준으로 등장하고 있다. HTML과 마찬가지로 XML은 SGML의 부분집합이라 할 수 있을 것이다. 하지만, HTML의 태그는 데이터 아이템을 어떻게 보여줄 것인가에 중점을 둔 것이라면, XML의 요소는 데이터 자체를 기술한다고 할 수 있을 것이다. 즉, XML은 자기 서술적인 특징을 바탕으로 XML 문서를 여러 형태로 보여줄 수 있고, 내용을 기반으로 데이터를 처리하여 어플리케이션의 목적에 맞게 재구성할 수 있다.

XML 문서는 데이터 구조를 나타내는 중첩된 요소의 집합으로 구성되어, 기존의 데이터베이스에서처럼 정해진 스키마를 갖고 있지 않지만 문서마다 그 구조를 갖고 있다고 볼 수 있다. XML의 데이터 모델은 구조상 기존의 데이터베이스와 많은 차이점이 있고, 기존 관계형 데이터베이스나 객체 지향 데이터베이스 질의어를 바로 적용하기에 부적합하다. 따라서 이러한 XML 문서들을 데이터베이스에 저장하는 방법들이 연구되어왔다.

XML 문서를 데이터베이스에 저장하는 방법에는 파일 시스템으로 저장하는 방법, 관계형 데이터베이

스 시스템에 저장하는 방법, 객체지향 데이터베이스 시스템에 저장하는 방법, XML 전용데이터베이스에 저장하는 방법 등이 있다[5].

파일 시스템은 XML 데이터의 저장이나, 문서 전체에 대한 검색은 용이하지만 문서의 갱신 등과 같은 특정한 질의의 처리에 대한 지원이 부족하다. 객체지향데이터베이스 시스템은 이상적으로는 XML 문서가 객체 기반에 구조를 두었기 때문에 XML 문서를 저장하기에 가장 적합한 저장방법이라고 할 수 있을 것이다. 하지만 현실적으로 대용량의 데이터베이스에 대한 복잡한 질의를 처리하기에는 어려움이 많다. 관계형 데이터베이스는 일반적인 데이터, 반 구조적 데이터가 공존할 수 있고, 대용량 데이터베이스에 대한 복잡한 질의의 처리에 효율적이기 때문에 현실적으로 XML 문서를 저장, 관리하기에는 효율적인 저장 방법 중에 하나라고 할 수 있을 것이다.[5~6]. XML 전용 데이터베이스는 XML에 가장 적합한 계층적 구조이고, XML 문서를 가장 적합하게 표현할 수 있기 때문에 가장 적합한 저장 방법이라 할 수 있을 것이다.

본 논문에서는 XML 문서를 데이터베이스에 저장 관리하는 좋은 방법을 연구하기 위하여, XML 문서를 저장하는 방법 중 관계형 데이터베이스에 저장하는 방법과 XML 전용 데이터베이스에 저장하는 방법을 비교 분석해 보았다. 앞으로 계속된 연구를 통하여 XML 문서를 효율적으로 저장할 수 있는 데이터베이스를 구현하는 연구를 계속해 나갈 것이다.

2. 본론

2.1 XML 데이터베이스의 필요성

웹 프로그램을 개발할 때 프리젠테이션 층, 프로그램 층, 데이터베이스 층으로 나누어 설계하게 되면 개발하기 쉽고, 유지·보수가 편리할 것이다. 프리젠테이션 층과 프로그램 층에 XML을 이용하는 방법에 관하여 많은 연구가 진행중이다. 데이터베이스 층 또한 많은 연구가 이루어지고 있으나, 기존의 관계형 데이터베이스를 이용하는 측면에서 언급되는 경우가 많았다. 하지만, 관계형 데이터베이스가 XML 문서의 구조를 그대로 보존하기 어려운 단점 때문에 XML 전용 데이터베이스에 대한 많은 연구가 이루어지고 있다. 프로그램에서 사용되는 모든 데이터를 저장하는 데이터베이스 층이 웹 프로그램에서도 매우 중요하다. 웹 프로그램 전체의 성능을 좌우하는 것은 XML의 특징을 어떻게 만드느냐에 달려있다고 할 수 있을 것이다. XML은 시스템간의 데이터 교환 형식으로서도 주목받고 있으며, 대규모의 기간 시스템을 중심으로 하는 EDI에서 인트라넷 부분 사이의 시스템간의 주고받음까지 간편하고 보다 효율적으로 데이터를 주고받는 수단으로 XML이 활용되고 있다.

이처럼 다양한 분야에서 사용되는 XML 문서는 단순한 문서의 교환뿐만 아니라 대용량의 처리를 위하여 데이터 보관은 매우 중요하게 되었으며, 다양하고 효율적인 저장 방법을 연구하고 있다.

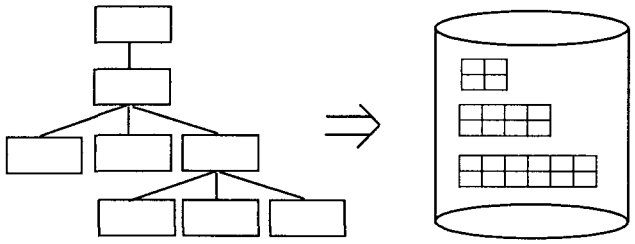
2.2 XML 문서의 관계형 데이터베이스로의 매핑

웹 프로그램이 등장하기 이전인 클라이언트/서버 시스템을 사용하던 시기에는 데이터베이스라고 하면 일반적으로 관계형 데이터베이스를 일컬었다 해도 과언이 아닐 것이다. 이후 객체지향 데이터베이스, 멀티미디어 데이터베이스 등 다양한 데이터베이스가 등장하였다. 하지만, 이들 데이터베이스는 대중화되지 못하고 있으며, 현재 가장 많이 사용되고 있고, 가장 접하기 쉬운 데이터베이스는 관계형 데이터베이스일 것이다. XML 문서를 저장하는 경우에도 관계형 데이터베이스에 저장하는 방법이 가장 처음 고려되었으며, 가장 활발한 연구가 이루어지고 있다고 해야할 것이다. 다음은 XML 문서를 관계형 데이터베이

스에 매핑하는 방법을 알아보았다.

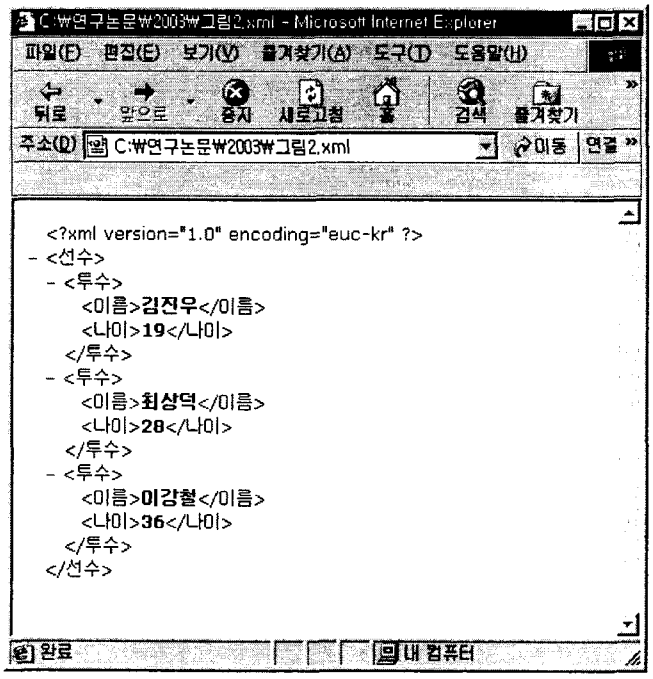
2.2.1 XML 구조의 테이블로의 매핑

관계형 데이터베이스는 행과 열의 테이블을 조합하여 표현하는 구조이기 때문에 트리 구조의 특성을 가지고 있는 XML 문서를 그대로 저장할 수는 없을 것이다. 그러므로 XML 문서의 구조를 복수개의 테이블로 나누어 XML 문서의 각 요소들을 테이블로 매핑(mapping)하는 방법을 사용하고 있다. [그림 1]은 트리 구조를 테이블로 매핑하는 방법을 나타내고 있다.



[그림 1] 트리 구조를 테이블로 매핑

관계형 데이터베이스에서 데이터를 추출하는 경우는 SQL을 사용하여 테이블에서 데이터를 추출하여 결합하고 이를 XML로 변환하는 것이 일반적이다. 관계형 데이터베이스에 저장되어 있는 테이블은 행과 열로 구성되어 있기 때문에 XML 문서가 일정한 수의 행 요소와 열 요소로 표현된 경우는 아주 쉽고, 보기 좋게 테이블에 매핑 할 수 있을 것이다. [그림 2]는 XML 문서를 테이블로 나타낸 것으로 XML 문서의 <투수>를 행으로, <이름>과 <나이>를 열로 하여 테이블로 매핑 하였다.



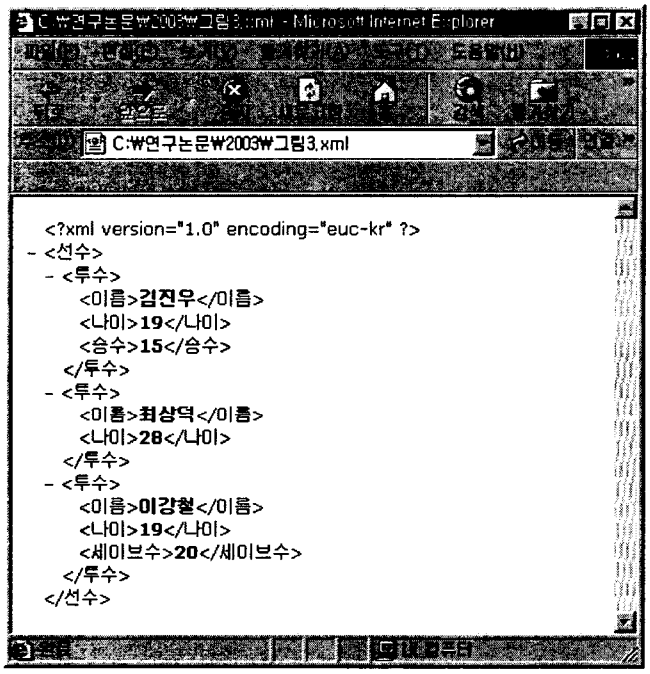
이름	나이
김진우	19
최상덕	28
이강철	36

[그림 2] XML 문서를 테이블로 매핑

2.2.2 선택적 요소에 대한 테이블 매핑

XML 문서에는 요소가 위의 [그림 2]에서처럼 모두 표현되는 경우도 있지만 선택적으로 표현되기도 한다. 이처럼 선택적으로 표현되는 경우 그 요소에 값이 있는 경우는 테이블에 표현되지만, 요소에 값이 없는 경우는 공백(null)으로 표현하게 된다. 즉, 선택적인 경우 요소의 값이 존재하는 것으로 테이블을 생성하여, 해당 요소에 그 값이 존재할 경우는 표현하고 존재하지 않는 경우는 공백으로 표현하면 된다.

[그림 3]은 [그림 2]의 XML 문서의 <투수> 김진우에게는 <승수> 요소의 값을 나타내고, 다른 <투수>에게는 <승수> 요소를 나타내지 않았으며, <투수> 이강철에게는 <세이브수> 요소의 값을 나타내고 다른 <투수>에게는 <세이브수> 요소를 나타내지 않았다. 즉, 선택적으로 특정 <투수>에게만 <승수> 요소와 <세이브수> 요소에 값이 존재하는 XML 문서를 테이블로 매핑 할 경우 모든 <투수>에게 <승수> 요소와 <세이브수> 요소가 있는 것으로 테이블을 생성하여, <승수> 요소에 값이 존재하는 김진우에게는 <승수> 요소의 값을 표현하고 다른 <투수>에게는 <승수> 요소에 공백(null)을 나타낸다. 마찬가지로 <세이브수> 요소에 값이 존재하는 이강철에게는 <세이브수> 요소의 값을 표현하고 다른 <투수>에게는 <세이브수> 요소에 공백(null)을 나타낸다.



⇒

이름	나이	승수	세이브수
김진우	19	15	null
최상덕	28	null	null
이강철	36	null	20

[그림 3] 선택적 요소가 있는 XML 문서의 테이블 매핑

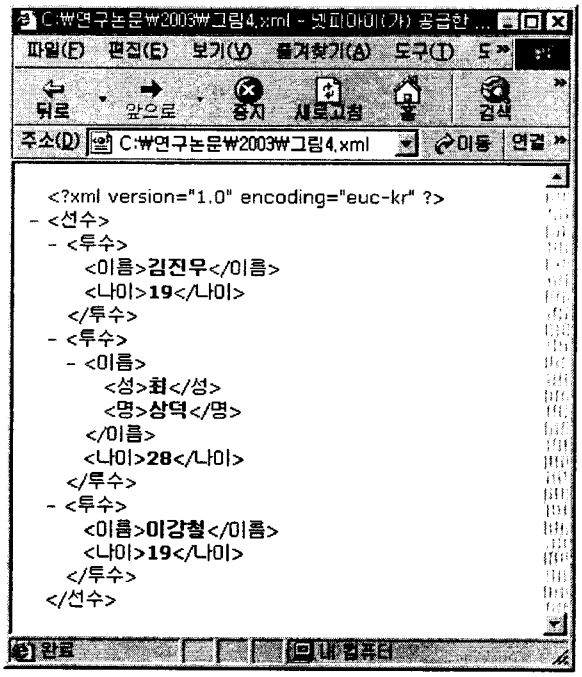
2.2.3 확장된 요소에 대한 테이블 생성

XML 문서에는 특정 요소를 확장하여 표현할 수 있다. 이런 경우 해당 요소를 몇 개의 부 요소(sub element)로 나누어 표현하기 때문에 테이블을 생성할 때 확장된 부 요소를 열에 포함하여 테이블을 생성한다. 확장된 요소에 대한 테이블을 생성할 때 확장된 요소의 부 요소에 값이 존재할 경우는 그 값을 표현하지만, 확장되지 않은 요소의 부 요소(sub element)는 공백(null) 값을 표현해야 하기 때문에 테이

블에 공백(null) 값이 너무 많이 나타나게 된다.

[그림 4]는 [그림 2]의 XML 문서에서 <이름> 요소의 값이 최상덕인 요소를 <성> 요소와 <명> 요소로 확장하여 표현하였다. 이를 테이블로 매핑 할 경우 <이름> 요소를 확장한 부 요소인 <성>과 <명>을 테이블의 열로 표현하여 나타내게 된다. 즉, <투수> 최상덕의 경우는 <이름> 요소를 확장하여 부 요소인 <성>과 <명>에 그 값을 나타내면 된다. 하지만 <이름> 요소를 확장하지 않은 <투수> 김진우와 이강철의 경우 <이름> 요소의 부 요소인 <성>과 <명>의 요소 값으로 공백(null)을 나타낸다.

관계형 데이터베이스에서는 [그림 4]의 테이블에서 성과 명을 이용하여 이름을 얻는다는 것은 간단하지 않다. 그렇기 때문에 일반적으로 테이블에 저장하기 쉽도록 XML 구조를 제한하여 표현하고, 그 외 구조의 XML인 경우는 미리 XLST를 사용하여 저장하기 쉬운 구조로 변경하여 사용한다.



⇒

이름	성	명	나이
김진우	null	null	19
최상덕	최	상덕	28
이강철	null	null	36

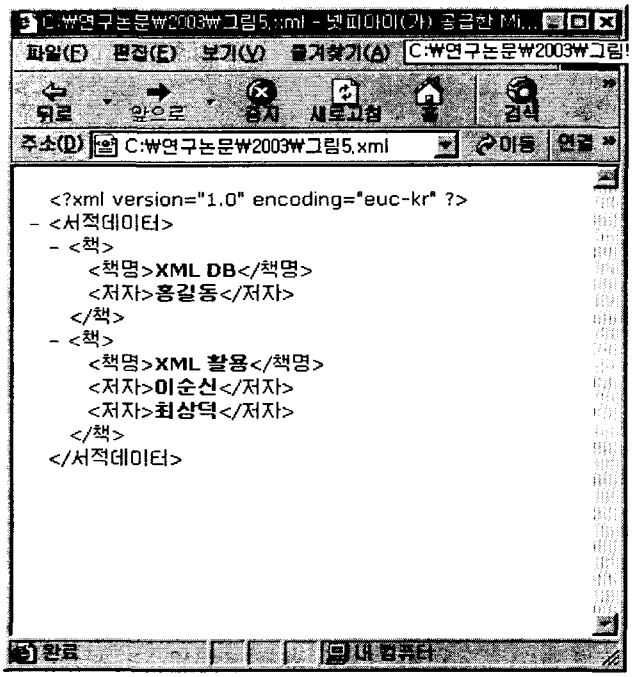
[그림 4] 확장된 요소가 있는 XML 문서의 테이블 매핑

2.2.4 복수값을 갖는 요소에 대한 테이블 생성

관계형 데이터베이스에는 테이블 설계를 효율적으로 하기 위해 사용되는 정규형이라는 것이 있다. 정규형에는 가장 유연한 규칙인 제1정규형(1NF)에서 가장 엄격한 규칙인 투사영 결합 정규형(PJNF)까지의 수준이 있는데, 엄격한 수준에 적합한 테이블 쪽이 갱신에 의한 데이터의 모순이 적고 불필요한 데이터가 적어진다. 일반적으로 테이블은 최소한 제1정규형의 규칙 「하나의 필드에는 하나의 값만이 들어간다.」에 따를 필요가 있다. 예를 들면, [그림 5]와 같이 서적 데이터베이스인 경우 1권의 책에 복수의 저자가 있을 수 있다. XML이라면 이와 같은 정보를 간단히 표현할 수 있을 것이다. 그러나, 관계형 데이터베이스에서는 [그림 5]와 같이 하나의 필드에 복수의 값이 들어 있는 테이블을 작성할 수는 없다. 이런 경우를 복합 개체(complex object)라고 한다. 복합 개체에는 {10, 50, 100} 과 같은 순서가 없는 값의 집합과, 〈김진우, 19세〉 와 같이 순서가 있는 값의 집합, 〈210,{<10, 250>,<20, 500>}〉과 같은 네스

트(nest) 관계의 객체 등이 있다.

XML과 DTD를 사용하면 하나의 XML 파일로 간단히 표현할 수 있으나, 관계형 데이터베이스로 표현하려면 복수 개의 열이나 행을 조합하거나 테이블을 나누는 등 구조가 복잡하게 되므로 많은 주의가 필요할 것이다. 다만, 최근의 관계형 데이터베이스에는 복합 개체를 지원하고 있는 것이 있어 어느 정도 복잡한 구조의 XML에도 대응할 수 있을 것으로 생각한다.



책명	저자
MXL DB	홍길동
XML 활용	이순신
	최상덕

[그림 5] 복수값을 갖는 XML 문서의 테이블 매핑

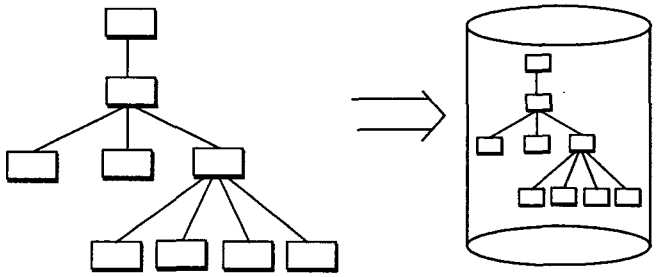
2.3 XML 전용 데이터베이스

XML 전용 데이터베이스란 XML 문서를 저장하기 위해 특별히 설계된 데이터베이스이다. 다른 데이터베이스와는 달리 트랜잭션, 보안, 다중 사용자 액세스, 프로그래밍 API, 쿼리 언어 등 많은 기능을 제공해 준다. 완전히 새로운 시점에서 XML 데이터베이스를 완성하는 예도 있으나, 많은 XML 전용 데이터베이스는 기존의 기술을 조합하여 각각의 장점만을 살린 혼합형으로 되어 있다.

XML 전용 데이터베이스의 특징은 XML과 데이터베이스 「데이터 모델」의 다른 점을 의식할 필요가 없다는 점이다. 관계형 데이터베이스는 XML과 다른 데이터 모델을 가지고 있다. 즉, 관계형 데이터베이스의 경우는 복수 테이블을 행과 열의 값에 의해 결합하는「관계형 모델」이 채용되었다. XML 전용 데이터베이스를 사용하여 XML을 XML 구조대로 저장할 수 있으면, 이와 같은 데이터 모델의 변환이 필요하지 않게 된다.

데이터베이스에서 데이터를 읽어내는 경우도 XML 전용 데이터베이스라면 관계형 데이터베이스 시스템을 이용한 경우와 같이 테이블이나 객체에서 XML을 재 구축할 필요가 없다. 다른 데이터 모델로의 변환이 필요하지 않은 경우의 또 하나의 장점으로, 설계 변경할 때에 수고를 더해야만 할 부분이 적다는 것을 들 수 있다.

XML 전용 데이터베이스는 XML 쿼리 언어 등과 같은 XML 관련 기능을 제공하므로 모든 메시지를 검색하는데 보다 빠르다. 반구조화 데이터를 저장하고, 어떤 상태 내에서의 검색 속도를 빠르게 하고, DTD를 갖지 않은 문서(Well-formed 문서)를 저장하는 등에 사용할 수 있다. XML 문서를 우선순위 구성과 상관없이 채택, 저장, 이해할 수 있다는 것이다. XML 문서내의 데이터를 관계형 또는 객체지향 데이터베이스로 전송하는 것은 매핑과 데이터베이스 스키마를 먼저 생성해야 하는 것을 필요로 한다.



[그림 6] XML 전용 데이터베이스에서의 XML 문서 저장방법

2.3.1 XML 전용 데이터베이스 구조

XML 전용 데이터베이스는 크게 두 가지 범주로 나눌 수 있다.

① 텍스트 기반 스토리지

텍스트 기반 스토리지는 전체 문서를 텍스트 형태로 저장하고, 문서를 액세스 할 수 있는 몇 가지 종류의 데이터베이스 기능을 제공해 준다. 이를 위한 간단한 방법은 관계형 데이터베이스 내에 문서를 BLOB형태로 저장하거나 또는 파일시스템 내에 일종의 파일로 저장하고, 문서에 대한 XML 인식 인덱스를 제공할 수 있다. 보다 나은 방법은 문서를 인덱스, 트랜젝션 지원 등의 기능을 갖는 최적화 데이터 저장소에 저장하는 것이다.

② 모델 기반 스토리지

모델 기반 스토리지는 DOM 또는 DOM의 변종과 같은 문서의 바이너리 모델을 기존의 또는 커스컴 데이터 저장소에 저장하는 것이다. 예를 들면 DOM을 엘리먼트, 애트리뷰트, 엔티티 등과 같은 관계형 테이블로 매핑할 수 있다. 이러한 기능을 수행하기 위해 특별하게 작성된 데이터 저장소에 미리 파싱된 형태로 DOM을 저장할 수 있다. 이는 공식적으로 'PDOM구현(Persistent DOM Implementations)'으로 알려진 특수한 목적의 XML 데이터베이스 역시 여기에 포함된다.

③ 텍스트와 모델 기반 스토리지의 차이점

구분	테스트 기반 스토리지	모델 기반 스토리지
라운드 트립(round-trip)	- 문서를 정확하게 round-trip	- 하부의 문서 모델 레벨에서만 가능
속도	- 전체 문서 또는 프레그먼트를 텍스트 형태로 리턴하는데 장점 - 문서크기, 파싱 속도에 의존적	- 서로 다른 문서로부터의 프레그먼트들을 결합하는데 장점 - 검색속도에 의존적

2.3.2 XML 전용 데이터베이스 기능

XML 전용 데이터베이스에서 제공해주는 주요 기능들로서 XML 데이터베이스의 특징과 향후 발전 방향에 대해 예측할 수 있을 것으로 본다.

① 다큐먼트 콜렉션

XML 전용 데이터베이스는 파일시스템에서의 디렉토리 개념과 유사하고, 관계형 데이터베이스에서의 테이블과 유사한 개념이라 할 수 있는 콜렉션 개념을 지원해 준다.

② 쿼리언어

XML 전용 데이터베이스는 하나 이상의 쿼리 언어를 지원한다. 수많은 독점적 쿼리 언어들도 지원해 주기는 하지만 쿼리 언어중 가장 많이 사용되는 것은 다중 문서에 대해 쿼리 할 수 있는 확장 기능을 갖는 XPath와 XQL이다. 쿼리의 범위는 전문검색에서부터 다중문서로 프레그먼트를 재결합하는 쿼리까지 다양할 수 있다. 향후 대부분의 XML 전용 데이터베이스에서는 W3C의 XQuery를 기본적으로 지원하게 될 것이다.

③ 갱신 및 삭제

XML 전용 데이터베이스는 문서를 갱신하고 삭제하는데 대한 다양한 정책을 가지고 있다. 많은 제품들은 XUpdate 언어를 지원하지만 , 문서의 프레그먼트를 변경할 수 있는 제품들은 자체적인 언어를 가지고 있다. 이러한 Update 기능은 아직까지는 많이 부족하기 때문에 계속해서 연구가 이루어져야할 기능중의 하나이다.

④ 트랜잭션 · 락킹 · 동시성

모든 XML 데이터베이스는 트랜잭션을 가상으로 지원하고, 락킹은 문서의 프레그먼트 레벨보다는 전체 문서 레벨에서 지원되고 있으므로 다중 사용자 동시성이 상대적으로 낮다. 이러한 이슈는 애플리케이션과 어떤 "문서(document)"로 구성되었는지에 따라 달라진다. 만약, 사용자 안내서가 개개의 챕터로 나누어져 있다면, 각각의 챕터가 문서가 될 것이고, 두 명의 작성자가 동시에 같은 챕터를 갱신하는 경우가 발생하기 힘들기 때문에 동시성 문제는 낮다고 할 수 있다. 반면 전체 회사를 위한 모든 고객의 데이터가 단일 문서 내에 저장되어 있다면, 문서 레벨 락킹은 좋지 않은 결과를 자져올 수 있다.

⑤ API

대부분의 XML 전용 데이터베이스는 프로그래밍 API를 제공한다. API는 일반적으로 ODBC와 유사한 인터페이스의 형태로 제공되며, 데이터베이스에 대한 연결, 메타 데이터 탐색, 쿼리 수행, 결과 가져오기 등의 메소드를 갖는다. 일잔적으로 XML 문자열, DOM 트리, 리턴 문서에 대한 SAX 파서나 XMLReader 등으로 결과 세트가 리턴 된다.

⑥ 라운드 트리핑(round-tripping)

XML 전용 데이터베이스의 중요한 기능 중 하나는 XML 문서 라운드 트립할 수 있다는 것이다. 즉, XML 문서를 XML 전용 데이터베이스 내에 저장하고 같은 문서를 그대로 복원하여 다시 가져올 수 있다. XML 전용 데이터베이스는 요소, 속성, PCDATA, 문서 순서 레벨에서 문서를 라운드 트립 할 수 있다. 얼마나 많은 라운드 트립을 할 수 있는지는 데이터베이스에 따라 다르다.

⑦ 리모트 데이터

일부 XML 전용 데이터베이스는 데이터베이스 내에 저장되어 있는 문서 내에 리모트 데이터를 포함할 수 있다. 일반적으로 ODBC, OLE DB, JDBC 등을 이용하여 관계형 데이터베이스로부터 가져온 데이터이고, 테이블 기반 매핑 또는 객체 관계형 매핑을 이용하여 모델링 된다. 대부분의 틱 전용 데이터베이스는 라이브 리모트 데이터를 지원하게 될 것이다.

⑧ 인덱스

대부분의 XML 전용 데이터베이스는 요소 및 속성값의 인덱싱을 지원한다. 비 XML 데이터베이스 내에서 처럼 인덱스는 쿼리의 속도를 빠르게 하기 위해 사용된다.

⑨ 외부 엔티티 스토리지

XML 전용 데이터베이스는 XML 문서를 저장할 때 외부 엔티티 처리시 확장하여 문서의 나머지 부분과 함께 저장되도록 할 것인가? 엔티티 참조를 남겨둘 것인가? 에 대한 결론은 없다. XML 전용 데이터베이스에서는 외부 참조 엔티티를 경우에 따라 확장할 것인지의 여부를 명시하고 있다.

2.3 관계형 데이터베이스와 XML 전용 데이터베이스의 비교

아래의 <표 1>은 관계형 데이터베이스와 XML 전용 데이터베이스를 비교 분석한 표이다.

<표 1> 관계형 데이터베이스와 XML 전용 데이터베이스의 비교

구분	관계형 데이터베이스	XML 전용 데이터베이스
계층적인 naming 구조	- 계층적 naming 지원하지 않음 - 데이터의 정규화 또는 단순화 필요(데이터의 효용성 제한) - 모든 계층의 naming 구조는 관계형 데이터베이스 위에 만들어져야 함	- XML에 가장 적합한 계층적 구조 - XML 문서를 네이티브하게 저장
계층적 트리구조 검색	- 트리 내 이동은 트리 구조를 따라 올라가고 그곳에서 다른 쪽으로 내려가는 방법으로만 가능 - intensive join의 계산 필요(성능저하 원인)	- XML 계층 구조에 적합한 Xpath, Xquery 사용 - 트리 구조에 따른 자유로운 검색 가능(빠르고 효율적인 검색)
스키마 변경	- 스키마의 변경에 따라 정규화를 다시 해야함	- 쉬운 스키마 변경
가변 길이와 형식을 갖는 데이터의 지원	- 데이터를 고정된 필드에 저장 - 고정된 형식과 길이를 갖는 데이터 필요	- XML 객체나 요소들을 원래의 형태 대로 저장 - 저장 시에 어떤 문제도 없음.
전문 검색/쿼리	- 데이터가 잘 정의된 형식과 필드 길이를 갖기를 요구 - 새로운 형태의 풍부한 다양성을 다룰 수 없음 - 데이터를 BLOB로 저장할 수는 있지만 데이터의 검색이나 질의, 인덱싱이 어려움	- 저장 매체로부터 데이터를 얻어내기 위한 전문검색 및 질의(Full- text Search/Query)가능
라운드 트리핑 (Round-tripping) XML 문서 구조를 그대로 복원할 수 있는 기능	- 요소의 순서, 속성 등 XML 문서의 구조를 그대로 보존하기 어려움	- 완벽한 Round-tripping 지원

3. 결론

XML 문서는 다양한 분야에서 다양한 방법으로 처리되고 있다. 이러한 XML 문서를 데이터베이스에 저장하는 방법에는 파일 시스템으로 저장하는 방법, 관계형 데이터베이스에 저장하는 방법, 객체지향 데이터베이스에 저장하는 방법, 반 구조적 데이터를 위한 고유 시스템인 XML 전용 데이터베이스에 저장하는 방법 등 다양한 방법이 있다.

본 논문에서는 위의 다양한 데이터베이스에 저장하는 방법 중에서 XML 문서를 관계형 데이터베이스에 저장하는 방법과 XML 전용 데이터베이스에 대해서 알아보고 이들을 비교 분석해 보았다. XML 문서를 관계형 데이터베이스에 저장하는 방법에는 일반적인 XML 문서를 테이블로 매핑하는 방법과 XML 문서의 선택적 요소를 테이블로 매핑하는 방법, XML 문서의 특정 요소를 확장했을 경우 확장된 요소에 대한 테이블 생성 방법, 그리고, 복수 개의 값을 갖는 요소에 대한 테이블을 생성하는 방법 등 다양한 방법으로 처리되고 있으며 앞으로도 계속 연구되겠지만 XML 문서 구조와 동일하게 처리되는 XML 전용 데이터베이스로의 변환보다는 그 효율성이 떨어질 것으로 본다. 다만, 아직까지는 XML 전용 데이터베이스의 안정성과 기존의 관계형 데이터베이스를 이용하는 많은 고객에 대한 신뢰성을 주지 못하고 있음은 사실이다. 그러므로 계속된 연구를 통하여 보다 더 나은 XML 데이터베이스가 될 수 있도록 해야할 것이다.

참고문헌

[1] 강민호, "실무예제 중심으로 엮은 XML", 21세기사, 2001.
[2] 김은중, "JSP와 XML", 도서출판 대림, 2001.
[3] 김채미, "전문가와 함께하는 XML Camp", 마이트Press, 2001.
[4] 이경하, "실전 XML 데이터베이스 구축", 성안당, 2002.
[5] 이형배, "이형배의 XML", (주)사이버출판사, 2002.
[6] 최효진, "PROFESSIONAL XML DATABASES", 정보문화사, 2001.
[7] 박상원, 정재목, 정태선, 김형주, "XML과 데이터베이스", 정보과학회지, p24~30, 2001.
[8] 신병주, 진민, 정민수, "XML 문서의 관계 데이터베이스 저장", 한국정보처리학회, p55~58, 2001.
[9] 이제민, 민경섭, 박상원, 김형주 "관계형 데이터베이스 시스템에서 XML-뷰를 통한 XML 데이터의 지원", 정보과학회논문지, 컴퓨팅의 실제 제7권 제3호, p202~210, 2001.
[10] 정태선, 박상원, 한상영, 김형주 "XML 데이터를 위한 객체지향 데이터베이스 스키마 및 질의 처리", 정보과학회논문지, 데이터베이스 제29권 제2호, p89~98, 2002.
[11] D. Florescu, D. Kossmann, "Storing and Querying XML Data using an RDBMS", Data Engineering Bullentin, Vol 22, No. 3, 1999.
[12] R. Bourret, "Mapping DTDs to Databases", www.xml.com/pub/a/2001/05/09/dtdtodbs.html, 2001.