3계층 XML 문서 저장 시스템의 설계

오준환*, 이병욱*
*경원대학교 전자계산학과
e-mail:linus@web.kyungwon.ac.kr

Design of a 3-Tiered XML Document Storage System

Jun-Hwan Oh*, Byung-Wook Lee*
*Dept of Computer Science, Kyungwon University

요 약

XML 문서와 같은 구조적 문서는 관계형 데이터베이스에 저장하는 것이 적합하다. 본 논문에서는 XML 문서의 각 엘리먼트를 관계형 데이터베이스에 검색을 위해 적정 노드까지만 깊이 우선 탐색 순서쌍에 의해 저장하고, 검색된 문서의 재생성 속도를 향상하기 위해 문서전체를 저장하는 방법을 제시하였다. 또 위에서 제시한 방법을 저장 시스템과 분석 검색하는 시스템을 서로 다른 사이트로 분리하는 것을 제안한다. 이를 통해 XML 문서를 서로 다른 사이트로 분리함으로써 서버의 부담을 줄여 저장 및 검색 성능을 향상한다.

1. 서론

HTML(Hyper Text Markup Language)은 사용자가 레이아웃을 임의로 지정할 수 없고 구조적 정보를 저장하고, 표현하기에 적합하지 않은 단점이 있다. SGML(Standard Generalized Markup Language)은 다양한 문서 형식을 지원할 수 있는 국제 표준문서이다[1]. 그러나 SGML은 마크업 규칙이 복잡하고 하이퍼텍스트를 지원하지 않기 때문에 인터넷에서 이용하기 부적합하다[1].

W3C에서는 이런 단점들을 극복하고자 1997년 12월에 새로이 XML(eXtensible Markup Language)을 제안하였다[2][3][4]. XML은 SGML에서 규칙들을 단순화하고 사용하지 않는 부분들을 제거하여 쉽게 어플리케이션을 개발 할 수 있게 하였다. 문서 구조의 특징이 태그에 의해서 분리되어 있고 구조를 나타내는 부분과 문서를 표현하는 양식이 분리되어 데이터베이스에서 질의를 가하여 문서를 검색할 수 있다. 기존의 HTML을 확장 보완하고 SGML을 간소화 한 것이기 때문에 웹 상에서의 다양한 형태의 전자 문서를 표현 할 수 있다. 좀더 다양하고 대용량의 문서들을 만들게 되면서 XML 문서의 효율적인 검색과 저장을 위해 객체지향 데이터베이스, 관계형 데이터베이스 등에 적용하는 연구가 추진되어 왔다[5][6][7][8].

본 논문에서는 XML 문서의 각 엘리먼트를 관계형 데이터베이스의 테이블에 저장하는 방법과 XML 문서를 저장하는 사이트와 XML 문서를 분석하고 검색 후 재 생성하는 사이트를 분리하여 서버의 부담을 덜어주는 방법을 제시한다.

2. XML 문서 저장 기법

XML 문서를 관계형 데이터베이스에 저장하는 방식이 다양하게 연구되어 왔다[4][7][8]. 아래에서는 이와 같은 구조를 갖는 문서의 저장시스템에 대한 관련 연구를 기술한다.

2.1 깊이 우선 탐색 순서쌍을 이용한 방법

깊이 우선 탐색 순서쌍 기법은 트리의 루트 노드로부터 깊이 우선 탐색 방식으로 노드를 방문하는 방식이다[4]. 노드의 방문순서 1쌍으로 이루어지는데 처음 방문할 때의 순서번호와 자신의 자식 노드의 방문이 모두 끝난 뒤의 노드의 방문 순서의 번호를 1쌍으로 구성한다. 이렇게 구성된 깊이 우선 탐색 순서쌍은 문서의 트리 구조상에서 특정 노드에 속해있는 하위 트리에 대한 질의를 하나의 SQL 문장으로 처리할 수 있다. 하지만 몇 단계 위 또는 몇 단계 아래의 특정 위치 엘리먼트에 대한 질의 처리는 깊이 우선 탐색 순서쌍 하나만으로는 어렵고 부가적인 속성 필드를 필요로 하거나, 추가적인 검색 및 연산이 필요하다. 또한 문서를 데이터베이스에 추가나 삭제 시 문서를 트리로 구성하여 깊이 우선 탐색 순서쌍 한 뒤에 저장해야 하는 작업이 필요하다.

그림 2.1은 논문 문서 형식을 XML 문서로 표현 예이다. 그림 2.2의 트리는 그림 2.1의 논문 XML 문서를 깊이 우선 탐색 순서쌍으로 변형한 형태의 예이다.

```
<논문>
<제목>XML문서 저장기법</제목>
<지은이>오준환</지은이>
<큰제목><큰제목번호>2</큰제목번호>XML저장 기법</큰제목>
<문서내용>
        <작은제목>리스트형태</작은제목>
        <단락>각 엘리먼트 ...<표번호>1</표번호>
        </단락>
</문서내용>
</논문>
```

그림 2.1 논문 XML 문서의 실제 예

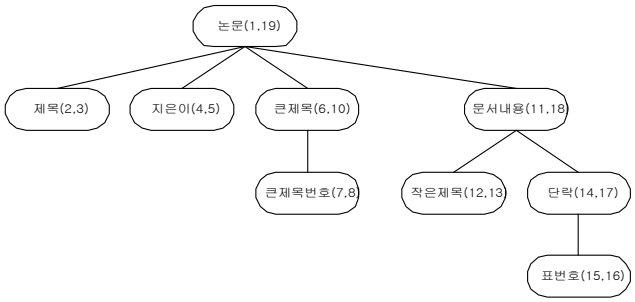

그림 2.2 논문 XML 문서의 깊이 우선 순서쌍

2.2 3계층 C/S 시스템의 장점

3계층 모델은 표현논리를 전단부 계층으로 하고 응용논리를 독립시키고, 자료논리와 데이터베이스 시스템을 후단부 계층으로 구성한 C/S모델이다[9][10]. 이 모델은 모든 기능들이 명확히 구분되어 있으므로 확장성이 좋고 강건하고 융통성이 뛰어나다. 처리능력이 대형화되어 조직체의 핵심업무를 처리하는데 사용된다. 3계층 모델은 다음과 같은 업무를 처리할 때 적합하다.

·전제 조직의 핵심 업무로 응용 분야가 다양할 때
·상이한 언어 또는 상이한 조직이 프로그래밍한 응용분야
·이질적인 데이터 소스를 이용한 경우

본 논문에서는 깊이 우선 탐색 순서쌍기법과 리스트구조 기법의 관련 연구를 이용하여 보다 효율적인 검색을 위한 XML 문서 저장 구조를 개선한다.

3. 3계층 저장 시스템의 설계

본 논문에서는 검색 후 XML 문서의 재생성의 속도를 향상시키는 방법을 제안하며, 저장관리 사이트와 검색 생성 사이트를 분리하는 3계층 저장 시스템을 제안한다.

3.1 문서의 중복 저장

구조적인 문서인 XML 문서를 저장하기 위해서는 구조적 정보인 엘리먼트에 대한 질의와 각 엘리먼트의 내용에 관한 질의가 가능해야 한다. 본 논문에서는 깊이 우선 탐색 순서쌍 저장 방식의 단점인 구조 검색 후 문서를 재 조합할 때의 생성 시간을 줄여 문서 결합을 개선하였다.

문서 검색을 위해 깊이 우선 탐색 순서쌍으로 테이블에 저장을 하여 마치 관계형 데이터베이스에서 테이블을 검색할 때 사용하는 인덱스처럼 사용하게 된다. 이 방법으로 저장을 할 때 문서 노드의 깊이는 관리자가 임으로 깊이를 정해준다.

깊이 우선 탐색 순서쌍을 이용하여 알맞은 문서를 검색 후 문서를 보여주기 위해 XML 문서 전체를 다른 테이블에 저장한다. 기존에는 각 엘리먼트 별로 분리해서 저장했기 때문에 재 생성하는 과정이 필요하였다. 하지만 제안하는 기법은 깊이 우선 탐색 순서쌍

을 인덱스처럼 사용하고 문서 전체가 다른 테이블에 저장 되어있으므로 재생성 없이 저장된 테이블에서 바로 불러올 수 있다. 아래 그림 3.1은 저장의 예를 보여준다.

깊이 우선 탐색 순서쌍을 위해 만든 테이블은 기존시스템 구조에 실제 문서가 저장되어 있는 문서를 가리키는 필드를 추가한다

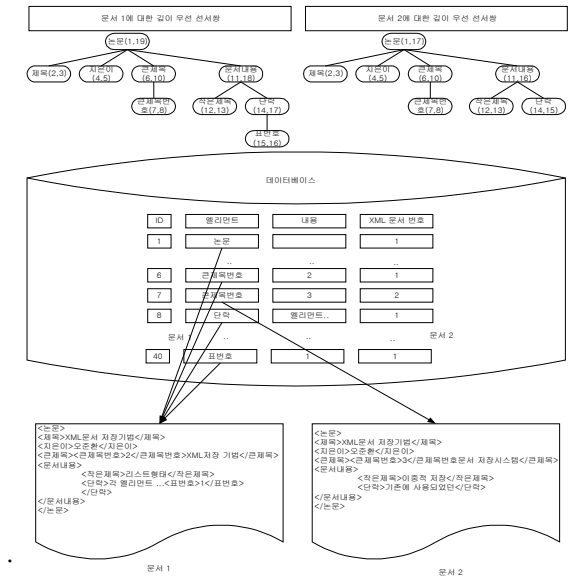

그림 3.1 시스템의 실제 예

3.2 XML 문서 저장 스키마

XML 문서는 XML 선언부 (XML Declaration), 문서 구조 정의부 (Document Type Definition), XML 실제 문서(Document Instance)로 구성된다[3][4]. 선언부는 문서의 규칙 등을 저장하는 곳이고, DTD는 문서의 구조적 정보를 정의하는 곳이고, 실제 문서는 DTD의 구조적 정보를 토대로 실제 내용이 들어있는 부분이다. 그림 3.2는 DTD의 예를 보여주고 있다. 이러한 특성을 가진 DTD를 이용하여 XML 문서를 각각 관계형 데이터베이스의 테이블에 사상을 한다.

```
<!DOCTYPE DOCUMENT [
<!ELEMENT OCUEMTN (CUSTOMER)*>
<!ELEMENT CUSTOMER (NAME, DATA,)*>
<!ELEMENT NAME (#PCDATA)*>
<!ELEMENT DATA (#PCDATA)*>
]>
```

그림 3.2 DTD의 실제 예

XML 문서의 속성을 관계형 데이터베이스에 적용하기 위해 XML TYPE을 관계형 데이터베이스의 TYPE으로 변경한다. CDDATA, PCDATA, ENTITY, ID 등은 스트링으로 변경한다.

저장 스키마에 나타나는 각 테이블의 특성은 다음과 같다.

- **tb_Element** : XML 문서의 내용 검색을 위한 엘리먼트 값을 저장 관리하는 테이블
- **tb_Attribute** : XML 문서의 내용 검색을 위한 속성 값을 저장 관리하는 테이블
- **tb_XMLDOC** : 전체의 XML 문서를 저장 관리하는 테이블.
- **tb_Elecontent** : Element 테이블에 저장된 정보를 참조로 하는 실제 문서의 내용을 저장 관리하는 테이블.

tb_Element 테이블은 tb_XMLDOC 테이블로부터 상속되어 각각 기본키와 외래키가 지정되어 있다. tb_Attribute 테이블과 tb_Elecontent 테이블의 관계도 마찬가지이다.

tb_Attribute 테이블은 tb_Element 테이블에 상속되어 속성을 가지고 있는 엘리먼트의 속성이름을 저장한다. 이 테이블 또한 tb_XMLDOC를 이용해 검색을 할 때 참조로 하는 테이블이다.

tb_Elecontent 테이블의 ATT_NAME 필드는 엘리먼트의 어트리뷰트가 있을 경우에 생성되는 필드이다. 엘리먼트의 어트리뷰트가 여러 개일 경우에는 ATT_NAME 필드가 그만큼 생성된다. ATT_NAME의 필드 명은 DTD에서 정의한 해당 엘리먼트의 어트리뷰트 이름과 동일하다.

tb_XMLDOC 테이블은 XML 문서전체를 저장하는 테이블이다. 일반 키워드 검색이 이 테이블을 이용한다. 엘리먼트와 속성을 짤라 따로 테이블에 저장한 기존방식이 아니라 이 테이블의 content 필드에서 전체문서를 불러오는 방식이므로 문서를 재 조합하는 시간을 단축할 수 있다.

본 논문에서는 XML 문서를 저장하는 방법을 제안하는 것이기 때문에 DTD저장에 관해서는 언급하지 않았다. 하지만 문서의 종류의 양이 많아지면 이 또한 관리를 해야 한다. 본 논문에서는 DTD전체를 XML 문서 전체를 저장하는 거와 같이 한 테이블에 저장을 한다.

표 3.1, 표 3.2, 표 3.3, 표 3.4는 위에서 정의 된 테이블의 구조를 보여준다.

표 3.1 tb_Element 테이블

tb_Element 테이블		
E_ID	INT(PK)	
DOC_ID	INT(FK)	
TAG	CHAR(40)	태그의 이름
LEVEL	INT	Tree위치

표 3.2 tb_Attribute 테이블

tb_Element 테이블		
A_ID	INT(PK)	
E_ID	INT(FK)	속성을 가지고 있는 에리먼트의 아이디
A_NAME	CHAR(40)	속성의 이름

표 3.3 tb_XMLDOC 테이블

tb_XMLDOC 테이블		
DOC_ID	INT(PK)	
CONTENT	XMLTYPE	XML 문서

표 3.4 tb_Elecontent 테이블

tb_elecontent 테이블		
ID	INT(PK)	
E_ID	INT(FK)	
E_NAME	CHAR(40)	
CONTENT	XMLTYPE	엘리먼트의 내용
ATT_NAME	XMLTYPE	속성의 내용

3.3 검색 방법

검색은 2가지 방법이 있다. 첫 번째 방법은 간단한 키워드 검색이고, 두 번째 방법은 엘리먼트 검색이다.

첫 번째 방법의 단순 키워드 검색은 기존의 일반 문서 검색에서와 비슷한 방법이다. 이는 3.2절에서 정의된 저장 테이블 중에 tb_XMLDOC라는 문서 전체를 저장하는 테이블에서 검색하는 방법이다. 하지만, DTD에 정의된 엘리먼트를 문서의 내용으로 오류검색을 할 수도 있다. 이런 오류를 줄이고 검색의 정확도를 높이기 위해 우선 tb_XMLDOC 테이블에서 검색을 한 후에 검색 결과물을 엘리먼트와 속성이 정의된 테이블을 이용해 결과물을 재분석한다. 재분석은 문서의 실재 내용에 있는 검색 결과와 엘리먼트, 속성이름을 검색한 결과를 분리하는 작업을 말한다. 검색된 문서 중에 문서의 실재 내용에는 키워드가 없고 엘리먼트와 속성이름에만 키워드가 있는 검색결과는 제외시킨다

두 번째 방법의 엘리먼트 검색은 DTD에 정의된 엘리먼트를 이용해 검색을 하는 방법이다. 이 방법은 문서 전체에 대해 검색이 아니라 tb_Elecontent 테이블에 저장되어 있는 자료를 이용해 검색하는 방법이다. 이 방법은 XML 문서를 저장할 때 문서 전체를 나누어 저장을 했을 경우에 어느 정도 내용 검색까지도 지원한다.

3.4 3 계층 XML 문서저장 시스템

기존에 연구되었던 방법은 저장 시스템과 검색, 생성 시스템이 동일 사이트에 있어서 단일서버에 부담을 많이 주었다. 제안된 시스템은 검색과 문서 재생성과 같은 업무 논리(business logic) 계층과 순수하게 데이터만을 저장하는 자료 접근(data access) 계층으로 분리하였다.

업무 논리 계층은 표현(presentation) 계층과 자료 접근 계층과의 연결 다리이다. 업무 논리는 작업을 수행하기 위해 표현 계층의 요청을 받아 응답을 한다. 또 표현 계층에서 받은 요청을 응답하기 위해 자료 접근 계층으로부터 자료를 요청해 응답을 받는다. 응답을 받은 자료를 업무 논리에 있는 룰이나 프로시저에 적용하여 표현 계층에 응답을 해준다. 이 로직에 있는 룰이나 프로시저들은 단지 표현 계층에만 작업 요청 및 응답을 하는 것이 아니라 필요하다면 업무 논리에 정의 된 다른 룰이나 프로시저들에게 작업의 요청 또는 응답을 해준다. 룰은 태스크의 흐름을 제어하는 수단이다.

이 시스템에서는 문서를 검색할 때 본문의 내용인지 엘리먼트나 속성의 이름인지를 판단하는 것이다. 프로시저는 XML 문서를 저장하기 위한 질의를 만드는 역할을 하고 문서 검색을 위한 질의를 만들고 그 질의에 의해 검색된 문서를 룰에 적용하여 오류 검색된 문서들을 제거하거나 룰에 적용해 엘리먼트 검색을 한다.

자료 접근 계층은 XML 문서 데이터를 업무 논리의 요청을 받아 정의, 접근 등을 한다. 이 계층은 데이터를 순수하게 저장하는 기능을 가지고 있으므로 표현 계층에서 바로 접근할 수 없도록 주의 해야한다. 이 계층은 같은 환경의 단일 데이터베이스나 분산 데

이터베이스들의 집합일 수도 있고, 다른 환경의 데이터베이스이거나 메인프레임의 집합일 수도 있다. 하지만 제안하는 시스템은 다른 관계형 데이터베이스만을 지원하기 위해 제안한다.

본 논문에서는 저장 시스템을 설계하는 것이기 때문에 3계층 중에 사용자와 관리자가 사용하는 표현 계층은 제외한 업무 논리 계층과 자료 접근 계층만을 제안한다. 시스템의 전체적인 구조는 다음과 같다.

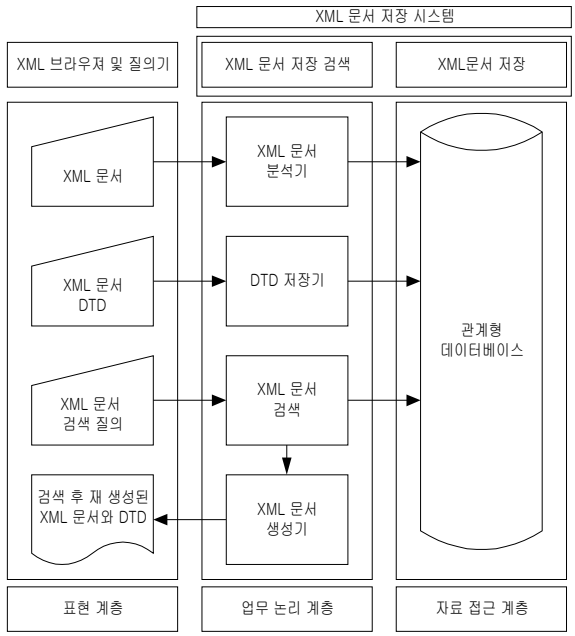

그림 3.3 3계층 XML 문서저장 시스템

3.5 성능 평가

성능 평가 요소는 재생성속도이다. 평가를 할 때 문서의 개수와 엘리먼트의 개수 당 생성속도 측정을 한다.

그림 3.4는 평가를 위한 기존 시스템과 제안한 시스템 각각의 문서 재생성 속도 측정 수식이다. 문서의 재생 속도 측정을 위한 평가 요소들은 가정할 요소들은 그림 3.5와 같다.

기존 시스템=$(R_C + R_T + (E_B \times B_T)) \times E_N$

제안 시스템=$R_T + E_N \times E_B \times E_N$

그림 3.4 평가 수식

기존 시스템의 성능평가 수식은 엘리먼트 하나가 호출되는 시간에 생성되는 시간을 더하고 엘리먼트의 개수만큼 곱해서 나온 수식이다. 제안된 수식은 엘리먼트의 생성시간이 없으므로 문서가 호출되는 시간만을 고려한다. 하지만 호출되는 시간이 엘리먼트의 양이 많아질 경우 크기가 커지므로 크기를 고려해 호출시간을 나타냈다.

레코드를 불러오는 시간 : R_T

불러온 레코드당 조합 시간 : R_C

바이트당 불러오는 시간 : B_T

엘리먼트 당 평균 바이트 : E_B

문서의 엘리먼트수 : E_N

그림 3.5 가정 요소

R_T=3ns, R_C=6ns, B_T=2ns, E_B=30byte라 가정할 때 문서의 엘리먼트 수를 각각 20개, 50개, 80개, 100개, 120개, 200개, 300개에서 각각 계산한다.

표 3.5 성능평가

엘리먼트 수(개)	기존시스템	제안시스템
20	1380	1203
50	3450	3003
80	5520	4803
100	6900	6003
120	8280	7203
200	12800	12003
300	20700	18003

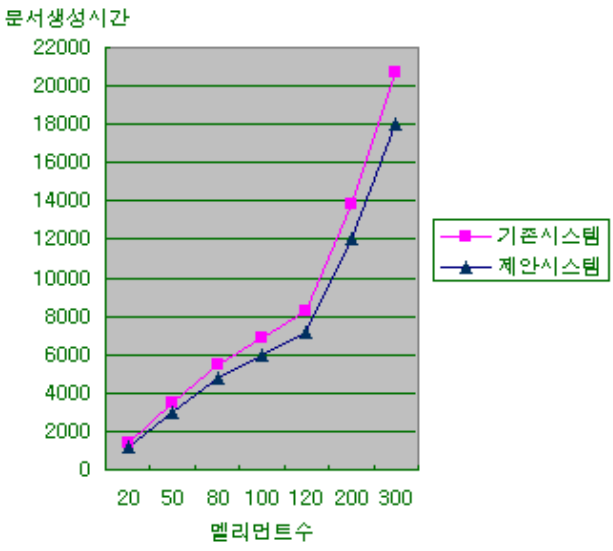

그림 3.6 성능 평가

그림 3.6에서 보는 바와 같이 제안 시스템이 엘리먼트의 수가 적을 때 보다 120개 이상이 되는 문서부터 성능이 향상되었다.

엘리먼트를 20에서부터 300개 사이로 한 이유는 작은 HTML 문서의 경우에도 문서내의 엘리먼트 수는 20개 이상 된다. 또 300개의 경우는 이 시스템은 엘리먼트의 개수가 많아지면 성능의 차이가 현저히 좋아지지만 현실에 맞추어 볼 때 엘리먼트의 수가 300이상의 경우의 문서는 많이 않기 때문에 300이상의 문서의 경우는 고려하지 않았다.

성능 향상의 이유는 엘리먼트의 수가 많을수록 제안한 시스템은 기존 시스템과는 달리 각각의 엘리먼트와 속성들을 테이블에서 불러 조합하는 시간이 없으므로 그 만큼 시간이 줄어든다. 엘리먼트가 늘어남에 따라 기존 시스템은 늘어난 엘리먼트를 조합하는 시간이 그 만큼 증가하게 된다. 따라서 제안한 시스템은 엘리먼트를 조합하는 시간이 없으므로 시간이 단축된다. 하지만 엘리먼트가 커지면 그 만큼 저장된 자료의 크기가 크므로 불러오는데 많은 시간이 걸린다. 하지만 이 시간은 엘리먼트를 조합하여 문서를 생성하는 시간이 아니라 문서를 불러오는 순수한 시간이므로 만약 문서의 크기를 반영하지 않고 불러온 문서를 재생성 하는데 걸리는 시간만을 반영한다면 그 차이는 엘리먼트의 크기에 따라 배 이상의 차이가 된다.

4. 결론 및 향후 연구 방향

관계형 데이터베이스에 저장하는 기존 방식은 각 엘리먼트로 나누어 저장하였다. 이 방법은 검색 후 문서를 재 생성할 때 생성시간이 많이 소요가 되어 효율이 떨어진다. 따라서 내용검색을 위한 엘리먼트들은 기존의 깊이 우선 탐색 방법으로 저장하고 검색 후 재생성을 위해 CLOB에 문서 전체를 저장하여 재생성 성능을 개선하였다. 또한 이 시스템은 데이터베이스 서버에는 자료만을 저장을 하고, 어플리케이션 서버를 두어 그곳에 그 외의 일들인 문서의 분석과 검색 후에 자료를 조합하여 문서를 만드는 작업 등을 수행한다.

본 논문에서 제안한 시스템은 기존 시스템과 달리 서버의 부담이 적어지므로 성능향상을 이루었다. 재생성 효율의 증가와 3계층 시스템의 설계로 인하여 이 시스템은 기존의 시스템보다 검색 후 재생성 속도가 향상되었다.

이 시스템의 단점은 재생성 속도를 향상시키기 위해 저장 공간이 중복된 양만큼 커지는 결과를 가져왔다. 그러나 저장 매체의 가격이 현저히 저렴하므로 이 문제는 무시할 수 있다. 재생성 속도나 검색 속도와 및 정확도를 향상하는 것이 중요하다.

향후 연구 방안은 제안한 방식인 3계층 저장 시스템이므로 이 장점을 이용해 관계형 데이터베이스뿐만이 아닌 객체지향 데이터베이스나 객체관계 데이터베이스에도 적용을 하는 것이다.

참고문헌

[1] ISO 8879, "Standard Generalized Markup Language(SGML)", Generva Sritzerland, 1986
[2] Extensible Markup Language(XML),"HTTP://www.w3.org/TR/PR-xml-971208"
[3] Frank Boumphyrey 외 11인, “Professional XML Applications", Wrox, 1999
[4] Steven Holzner, "XML Complete", The McGraw Hill Compaines, 1999
[5] 이용석, 손기락, “XML 문서 저장 시스템 설계 및 구현”, 석사학위논문, 한국외국어대학교, 1999
[6] 김영일, 김형선, “객체지향형 데이터베이스를 이용한 XML 문서 저장 시스템 설계”, 정보과학회 학술 발표 논문집(I), 26권 2호, 1999
[7] 류진영, “관계형 데이터베이스에서 XML 문서 저장 방안 연구”, 정보과학회 가을 학술발표 논문집, 1998
[8] 허명식, 손기락, “XQL을 지원하는 XML 문서저장 시스템”, 정보과학회 학술 발표 논문집(I), 26권 2호, 1999
[9] 이병욱, “클라이언트/서버 실습 중심 데이터베이스 시스템”, 생능출판사, 1999
[10] Korth, Silberschatz, "Database System Concepts", The McGraw Hill Compaines, 1997