碩士學位論文

XML文書의 관계형 데이터베이스 게이트웨이 設計 및 具現

濟州大學校 大學院
情報工學科

洪 東 賢

2000年 12月

XML文書의 관계형 데이터베이스 게이트웨이 設計 및 具現

指導敎授 郭鎬榮

洪東賢

이 論文을 工學 碩士學位 論文으로 提出함

2000年 12月

洪東賢의 工學碩士學位 論文을 確認함

審査委員長 李 尙 俊

委 員 郭 鎬 榮

委 員 宋 旺 暻

濟州大學校 大學院

2000年 12月

The Design and Implementation of Gateway between XML Document and Relation-Database

Dong-Hyun Hong
(Supervised by professor Ho-Young Kwak)

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING

2000. 12

DEPARTMENT OF INFORMATION ENGINEERING
GRADUATE SCHOOL CHEJU NATIONAL UNIVERSITY

목 차

Summary

XML(Extensible Markup Language) which overcomes the defects of SGML and which adopts the advantages of HTML and SGML, appears in the Internet business world with a great interest. Because of these characteristics, XML is used in many application. Therefore, it is necessary that documents using XML should be efficiently stored and managed. So far, few of studies that stores XML documents in the relational database, not only have been performed, but they also cannot perfectly reconstruct the original XML documents from XML data stored in DB.

In this paper, using the characteristics of XML document, the method that separately stores the structures and contents of XML documents, is proposed for the relational database(RDB), which is representative of the industrial world.

In this study, the design of the storage architectures to efficiently store XML documents and easily recontstruct the original documents in RDB, is based on E R modeling. In order to practically store XML documents in RDB, XML Parser and Inserter that handle XML documents according to the proposed storage architecture, were implemented. And when the structures and the contents of XML documents are separately queried, to turn out a complete result, Extractor and XML Composer were also implemented. Moreover, the convenient user interface was designed to easily store and query XML documents.

Ⅰ. 서 론

현재 인터넷 상에서의 문서는 World Wide Web을 구성하는 표준 문서 양식인 HTML을 이용하여 많은 문서가 작성되고 있다. 웹의 발전은 인터넷의 판도를 바꿀만큼 그 영향이 컸는데 웹의 발전에 가장 큰 공헌을 한 것중의 하나가 바로 HTML(Hyper Text Markup Language)이다. HTML은 하이퍼 텍스트 기능을 지원하고, 누구나 사용할 수 있을 만큼 편리하였다. 그러나 HTML은 단순한 홈페이지 작업을 하기에 매우 편리하지만 기능적으로 살펴보면 화면상에 어떠한 형태로 보여지는 기능 외에는 별다른 기능을 제공하고 있지 않을뿐 아니라 고정된 태그만을 사용하고 있고, 또 페이지 레이아웃(Layout) 형태를 임의로 지정할 수 없다. 즉, 사용자가 쉽게 사용할 수 있다는 장점은 있지만 너무 단순하고 고정된 태그만을 사용하기 때문에 복잡한 응용분야에는 잘 적응하지 못하는 한계가 있다. 그에 비해 또 다른 마크업 언어인 SGML(Standard Generalized Markup Language)은 표현 능력이 뛰어나고 문서구조를 기반으로 한 다양한 응용에 사용할 수 있지만 문법이 너무나 복잡하여 실제 구현이 너무 어려운 단점이 있다.

SGML은 문서의 기술에 필요한 태그를 생성할 수 있고, 문서의 내용이나 구조를 정의할 수 있으며, 구조화된 문서 데이터를 상호 교환할 수 있고 문서의 내용이나 구조를 정의할 수 있다. 그러나 이 SGML은 구조화된 문서 데이터를 상호 교환할 수 있다는 장점을 지니고는 있지만, 마크업 규칙이 복잡한데다가 인터넷의 문서 또는 데이터의 교환을 위한 목적으로 만들어져 있지 않기 때문에 현재 웹에서 사용되고 있는 하이퍼링크의 기능을 제대로 사용할 수 없는 단점이 있

다.

이러한 이유로 사용법이 복잡한 SGML의 단점을 극복하고 HTML과 SGML의 장점만을 취한 언어인 XML(eXtensible Markup Language)이 인터넷 업계의 주요 관심사로 대두되었다.

어떤 자료에 의하면 현재 웹페이지의 75%이상이 데이터베이스의 자료에 의해서 생성된다고 보고되어 있다. 이 데이터들은 Server Side 어플리케이션에 의해서 HTML 문서들이 생성된다. 브라우저는 그러한 HTML 문서들을 랜더한다. 대안은 이 데이터들을 XML문서로 전환하고 클라이언트에서 실시간으로 시각적 변환을 위한 스타일 쉬트를 생성하게끔 하는 것이다.

이러한 접근은 다음과 같은 몇 가지 이점이 있다.

1. 보다 나은 디자인

이러한 접근방법은 데이터베이스적 측면과 비즈니스 로직, 디자인 레이아웃을 분리하게끔 한다. 이러한 다른 영역은 현재 HTML 태그와 SQL문이 삽입되어져 있는 C나 Perl과 같은 Server Side 웹 어플리케이션 사이에서 강력하게 연결되어 있다. 이 결과 어플리케이션의 유지와 확장이 어렵게 된다.

2. 다른 클라이언트의 지원

클라이언트 각자의 소프트웨어 또는 PDF나 RTF와 같은 프린팅 포맷에 대한 스타일 시트의 생성이 콘텐트로부터 분리된다.

3. 로드 밸런싱(Load Balancing)

컴퓨터의 로드가 서버와 클라이언트 사이에 나누어질 수 있다.

4. 낮은 대역폭

HTML문서는 XML과 스타일시트보다 평균적으로 훨씬 높은 대역폭을 요구한다.

XML 문서에서는 문서의 구조적 특성이 적절한 태그(Tag)에 의해서 분리되고 재현 양식에 관한 정보는 별도의 스타일 문서에 의해 제공되므로 마치 데이터베이스에 질의를 가하는 것처럼 문서를 검색할 수 있으며 트랙잭션 단위의 인트라넷 데이터의 표현도 용이하다는 장점을 가지고 있기 때문에 웹상에서의 다양한 형식의 전자문서를 묘사하기에 매우 적합하다. 또한 기존 HTML을 확장·보완하였기 때문에 HTML 을 그대로 사용하면서 더욱 복잡하고 다양한 구조적 문서를 작성할 수 있다.

XML 문서는 SGML의 다양한 구조의 문서를 작성할 수 있는 특징과 HTML처럼 웹상에서 쉽게 사용할 수 있다는 장점으로 인하여 향후 인터넷 문서의 표준으로 사용될 것으로 예측되며 다양하고 방대한 문서들이 생산될 것으로 보인다. 그에 따라 XML 문서의 양이 방대해지고 사용자는 원하는 XML 문서를 쉽게 검색하고 관리할 수 있기를 원한다. 그러기 위해서는 XML 문서를 데이터베이스에 저장하여 관리하여야 한다.

본 논문에서는 데이터베이스에 XML 문서를 저장하고 검색, 수정할 수 있는 방안에 대하여 연구하였다. 또한 XML 문서를 저장할 수 있는 XML 문서 저장 시스템을 구현함으로써 XML 문서를 쉽게 저장하고, 검색·수정할 수 있게 하였다.

Ⅱ.관 련 연 구

1. XML(eXtensible Markup Language)

XML은 플랫폼(platform)에 비의존적이며, 사용자 스스로 확장이 가능하고, 복합구조(Complex Structure)와 검증(Validation) 등 SGML의 기본 특징을 그대로 지원하며 구현하기도 쉬운 장점이 있다. 사용자가 원하는 대로 태그 세트와 속성을 지정할 수 있고, 새로운 태그세트를 만들 수도 있는 언어적인 성격이 강하다. 즉, HTML이 태그 닝어리였다면 XML은 확장이 가능하면서 자바스크립트(JavaScript)와 비주얼베이직(Visual Basic) 스크립트 등과도 연동이 가능한 개발도구로서의 특징이 있다.

XML을 활용할 수 있는 응용분야는 다음과 같다.

이기종(異機種) 데이터베이스 사이의 데이터 교환 : XML의 성격인 태그의 확장성, 중첩된 구조를 이용한 데이터베이스 정보의 표현 가능성, 그리고 자료가 올바른지를 검색할 수 있는 검증성을 이용해 이기종 데이터베이스간의 데이터 교환을 손쉽게 할 수 있다.

서버의 일처리 부하를 클라이언트로 분배 : XML은 각 웹 클라이언트를 이용해 서버가 해야 할 일을 분산 처리하는데 쉽게 되어 있다.

사용자에게 같은 데이터를 다른 형태로 보여주는 기능 : 예로,

각국의 언어로 된 메뉴얼을 한 문서에 담아 놓고 사용자가 원하는 언어로 볼 수 있도록 할 수 있다. 또는 하나의 전화 번호부가 이름 순서로, 때로는 직업 순서로 단 한번의 클릭으로 전환되는 문서 등을 들 수 있다. XML의 환경 아래에서 위의 작업은 XML 편집기를 이용해 구조화된 목차를 담고 있는 표준 XML 문서를 만들고, 이를 뷰어(Viewer)를 이용해 사용자에게 보여주고 동작하게 하는 모습으로 바뀌게 될 것으로 보인다.

개별적인 사용자에게 필요한 정보를 쉽게 발견할 수 있도록 도와주는 지능형 웹 에이전트(Agent) 인터넷에서 웹에이전트(웹서비스를 제공하는 프로그램)가 XML로 된 문서를 이용해 데이터베이스를 구축하고 정보를 정리하여 그 사람만을 위한 가이드를 수행하게 될 것으로 예측된다.

이와같이 XML은 다양한 응용분야의 요구사항을 충족시켜줄 수 있으므로 실제 응용 개발에서 많이 사용된다.

2. XML(eXtensible Markup Language)의 구조

XML 문서는 선언부, 문서형식 정의부(Document Type Definition: DTD), 문서부(Document Instance: DI)로 구성되어 있는데, 선언부는 SGML과 비교하면 상당히 축소되어 있고, DTD는 SGML의 형태를 많이 보존하고 있다. 문서부에 있어서 SGML과 크게 다른 것은 태그 최소화나 태그 생략이 적용되지 않는다는 것이다. 이것은 문서부에서 문서의 구조를 쉽게 파악할 수 있어서, 문서의 탐색

이나, 문서의 보관 및 관리에 상당히 유리한 점으로 작용된다. 본 논문에서는 그러한 XML 문서부를 현재 산업계에서 많이 사용되고 있는 관계형 데이터 베이스에 저장하는 방법을 제시하였다.

XML에서는 DTD가 문서의 논리적 구조를 나타낸다. 그리고 XML인스턴스 문서들은 DTD에 따라 논리적 구조를 띄게 된다. 따라서 이를 이용해 XML문서를 위한 데이터베이스의 논리구조 즉, XML 문서 데이터베이스 스키마를 만들 수 있다. 이렇게 DTD와 같은 구조적인 정보를 이용하여 데이터베이스를 설계할 때, 정보의 손실이 없도록 하는 동시에 범용적인 XML문서 데이터베이스를 구축하기 위해서는 다음과 같은 조건을 만족하여야 한다.

1) 모든 타입의 문서는 같은 데이터베이스에 저장할 수 있어야 한다.

2) 문서의 모든 데이터를 데이터베이스에 저장하여야 한다.

3) 전체문서가 데이터베이스로부터 재구성될 수 있어야 한다.

4) 새로운 문서를 데이터베이스에 들어 있는 다른 문서들의 일부분을 합쳐서 재구성할 수 있어야 한다.

5) 데이터 베이스에 저장되는 문서는 Well Formed 문서이어야 한다.

위의 조건을 만족하기 위해 서는 많은 연구가 필요하다.

3. XML(eXtensible Markup Language) 문서

1) XML 문서 유형

XML 문서는 적격문서와 유효문서로 나눌 수 있다. 만일 DTD를 구성해야된다면 유효한 문서를 구성하는 방법에 주의해야만 한다.

적격(Well Formedness)문서 : 어떤 문서가 하나의 XML 문서로 간주되는데 필요한 최소한의 필수조건들의 집합(이는 XML 사양에 명시되어 있다)을 의미한다.

유효(Vaild)문서 : 유효한 문서가 되려면 무엇보다도 XL 문서에 연관된 DTD를 지켜야 한다. 즉, DTD에 정의된 문서의 전반적 구조와 요소나 특성에 사용할 수 있는 데이터형들을 정확히 지켜야만 유효한 문서가 될 수 있다. 유효한 문서는 적격 문서보다 더 엄격한 조건들을 통과해야 한다. 정의에 따르면 유효한 문서는 또한 적격 문서이기도 하다.

2) XML 문서의 구성

(1) XML 선언 (XML Definition)

XML선언의 기본적인 형태는 처리지시(Processing Instruction) 요소인 <?이름....?>과 동일하다.

최소한의 형태의 XML 선언은 예약어 xml과 버전 번호로 구성된다. 문서 자체의 인코딩에 관한 특별한 사항이 있다면 인코딩 정보를 넣을 수 있다.

(2) 요소들(Elements)

XML 문서는 태그들로 마크업 된 데이터들로 구성된다. 요소란 각각의 시작 태그/종료 태그 쌍과 그 사이에 들어 있는 데이터를 통칭하는 것이다. 예를 들어, 어떠한 책이 장과 절, 문단 등으로 구성되어 있는 것과 마찬가지로 문서도 특정한 조각들이 결합된 구조를 가진다.

그러한 조각들을 요소(element)라고 한다.

1. 요소의 선언

어떠한 문서의 본문에 그 문서의 요소들이 적격성(well formedness)을 만족하는 형태로 배치되어 있다고 해도 요소들과 관련 특성들의 형(type)에 대한 선언이 없다면(DTD에 요소에 관련된 선언이 없다면) 그 요소들의 유효성을 점검할 수 없다.

어떤 하나의 요소가 다음과 같은 조건을 만족할 때 그 요소를 '유효하다(valid)' 라고 말한다.

DTD 안에 그 요소형에 대한 선언이 존재한다. 즉, 요소의 이름과 일치하는 요소 선언이 DTD안에 있어야 한다.

DTD 안에 그 요소형에 포함된 모든 특성들과 그 값의 형(데이터형 또는 다른 요소)에 대한 선언이 존재해야 한다.

그리고 문서에 나타난 요소의 내용의 데이터형이 선언에 정의된 내용 스키마의 데이터형과 일치해야 한다. 그리고 요소 선언의 경우 동일한 문서에 대해서 하나의 요소를 서로 다른 방식으로 한 번 이상 선언해서는 안 된다는 규칙이 존재한다.

2. 요소 내용 모델

요소의 선언에서 요소 이름 뒤에는 그 요소에 포함될 수 있는(꼭 포함해야 하거나 또는 생략해도 되는, 그리고 반복해서 나와도 되는) 종속적인 요소들의 목록이 나오는데, 그것을 내용 모델(content model)이라고 한다. 내용 모델은 요소선언을 이루는 한 부분이며, 요소선에서 괄호('(',')')로 둘러 쌓인 부분이 내용모델이다.

Table 1. A axample of element content model

```
<!ELEMENT sample (data. token, volume, substance)>
```

요소 sample은 data, token, volume, substance라는 부 요소들로 이루어진다. sample요소를 문서 인스턴스에서 사용하면 다음과 같은 구조가 되는데, 부 요소들의 나열 순서를 그대로 지켜야 한다.

내용 모델에 쓰인 "언어"는 유닉스에서 문서의 텍스트를 검색하거나 치환할 때 사용하는 정규 표현식(regular expression)과 비슷하다.

Table 2. Symbol of content model

Symbol	Usage
,	Order assignment
|	Choice
+	Repeat(Minimum 1)
*	Repeat
?	Omission possibility
()	Grouping

3) 특성(Attribute)

특성이란 XML 파서가 애플리케이션에게 보내는 값들을 뜻하는데, 요소의 내용과는 구별되는 값이다. 특성의 값은 요소의 시작 태그에서 지정한다.

(1) 특성목록의 선언

요소는 특성들을 가질 수 있다. 특성 목록의 선언은 <!ATTLIST 라는 지시로 시작하며 그 다음에는 특성들이 속할 요소의 이름과 특성의 이름, 그리고 그 특성에 넣을 수 있는 값의 데이터 형이나 값들의 목록이 나오고, 마지막으로 특성의 기본 값이 나온다. 하나의 요소가 가질 수 있는 특성의 개수는 제한이 없다.

적격문서의 경우 파서는 DTD를 참조하지 않으므로 특성값들을

그대로 문자 데이터로 취급한다. 그러나 유효한 문서의 경우 특성값들은 DTD에 정의된 방식에 따라 처리된다.

4) 개체(Entity)

개체는 문서를 만들 때 하나의 문서에서 긴 텍스트를 반복적으로 입력해야 하는 번거로움을 줄이기 위해 주로 쓰인다. C++의 매크로나 상수 같은 것이라고 할 수 있다.

개체는 내부와 외부로 구분되어지는데, 내부와 외부란 말은 개체 선언을 기준으로 한 내부와 외부를 뜻하며, 그 개체가 선언되는 XML 문서 인스턴스의 내부나 외부를 뜻하는 것이 아니다. 개체의 종류나 형식을 뜻한다.

Table 3. Example of Entity

```
<!ENTITY entityname "replacement word">
```


개체들을 여러 문서 인스턴스들이 공유해야 할 때 그것들을 하나의 외부 DTD안에 복사해 두고 사용할 수도 있겠지만 개체들을 여러 DTD파일들 사이에서 공유하는 경우에 치환용 텍스트 문서 인스턴스나 외부 DTD와는 개별적인 파일에 저장해서 공유할 수 있다면 시간이나 노력을 훨씬 절감할 수 있다. XML에서는 그러한 개체들을 외부 개체라고 한다.

외부개체를 선언할 때에는 PUBLIC키워드를 통해서 공용식별자(public identifier)를 지정한다. 파서는 우선 그 식별자를 이용해서 파일의 위치를 판단하며, 식별자로 판단할 수 없을 때는 다음에 나오는 파일경로 이름을 사용한다.

Table 4. Law of XML entity processing

문 맥	개체의 종류		문 자
	내부 범용	외부 해석된 범용	
내용안의 참조	포함됨	유효성 검사의 경우	포함됨
속성값 안의 참조	리터럴로 포함됨	허용되지 않음	포함됨
속성값으로 나타날 때		허용되지 않음	허용되지 않음
개체 값 안의 참조	건너뜀	건너뜀	포함됨
DTD안의 참조	허용되지 않음	허용되지 않음	허용되지 않음

5) 문서원형선언

<!DOCTYPE[...]>선언은 문서의 프롤로그에 나오는 XML선언 뒤에 나오며 문서에 필요한 모든 선언들이 담긴 세션을 지정하는 의미를 가진다. 적격문서가 개체들을 선언하려면 DOCTYPE 선언을 포함하는 XML 문서는 다음과 같은 형태를 가진다.

문서원형정의(DTD)는 문서의 모든 요소들과 그에 관련된 특성들에 대한 모든 선언들을 담는다. 또한 문서 문선원형선언 안에는 루트요소에 해당하는 요소의 선언이 반드시 포함되어야 한다. 즉, 문서원형의 이름은 그 문서의 루트 요소의 이름과 반드 일치해야 한다.

DOCTYPE 선언은 내부 부분집합과 외부 부분집합으로 구성된다. 하나의 DTD 선언이 외부 DTD파일을 참조함과 동시에 내부 선언들을 포함하는 것도 가능하다.

XML 파서는 내부 부분집합을 먼저 읽는다. 그리고 외부 부분집합과 내부 부분집합의 선언들 사이에 어떠한 모순이나 충돌이 있다면 내부의 것들을 우선하고, 외부의 것들은 무시한다. 내부가 외부에 우선하므로 표준으로 쓰이는 외부 DTD의 선언들 중 특정한 선언들을 덮어쓰는(overrride)것이 가능하다

(1) 문서 원형 정의

문서 원형 정의는 DOCTYPE 선언 안에 들어간다. 문서 원형정

의 는 문서를 설명하는데 필요한 모든 요소와 특성, 개체, 그리고 기타 선언들로 구성된다.

6) 처리지시(Processing Instruction)

XML이 범용적이고 확장성이 뛰어나긴 하지만, 순수 XML만으로 처리할 수 없는 경우가 있다. 그래서 XML에서는 다른 응용 프로그램의 능력을 사용할 수 있는 메카니즘이 존재한다. 외부 응용 프로그램의 기능을 사용하려 할 때 필요한 것이 바로 처리 지시(Processing Instruction)이다. 처리지시들은 문자 데이터로 취급되지 않으며, 변화없이 애플리케이션에게 넘겨진다. 파서는 문서를 처리하다가 발견한 처리 지시를 변경하지 않은 채 그 파서를 이용하는 애플리케이션에게 넘겨줘야 한다. 애플리케이션은 그 처리 지시를 이용해서 문서에 대한 특정한 작업을 수행한다.

Ⅲ. 관계형 데이터베이스에의 XML 문서저장기법 설계

1. 파서

XML 문서는 컴퓨터로 처리하기 위한 문서이다. 단순히 비트 혹은 문자들의 연속인 XML 문서가 컴퓨터에 의해 처리되기 위해서는 그 구조를 복원, 프로그램에 의해 처리 가능한 데이터 구조나 구조화된 정보의 흐름 등으로 변환하는 것이 필요하다. 이런 일을 해 주는 것이 XML 파서이고, 파서가 처리한 XML 문서를 응용프로그램에 전달하는 표준화된 파서 규약이 DOM과 SAX이다.

XML 문서는 요소(element), 속성(attribute), 텍스트 등으로 구성된 나무 구조의 계층화한 정보로 볼 수 있다. 따라서 이들을 각각 객체에 대응시키면 XML 문서를 트리구조의 객체로 표현 할 수 있다. 이와 같은 XML 문서를 나타내는 객체들의 인터페이스를 표준으로 정해 놓은 것이 DOM이다. 일반적인 DOM 파서는 바로 XML 문서로부터 DOM 트리 구조를 생성하는 역할을 한다.

반면 SAX는 XML 문서를 앞에서 뒤로 읽어 가면서 요소, 속성 등이 나타났는지 알려주는 구조 API이다. 즉, 문서의 구조를 일련의 사건(이벤트)흐름으로 표현한다고 볼 수 있다. 이런 이벤트들은 일종의 정보 덩어리로 볼 때, SAX파서는 XML 문서를 정보 덩어리의 흐름으로 바꾸는 도구이다.

파싱 후 문서를 처리하는 면에서의 DOM과 SAX의 주요 차이점은 DOM은 몇번이고 원하는 부분을 추가 및 수정할 수 있는 수단인데 비해, SAX는 문서를 처음에서 끝까지 순차적으로 처리한다는 것이다.

우선 DOM 뿐만 아니라 일반적으로 문서의 구조가 메모리에 있을 때 유리한 점은 다음과 같다.

1) 문서의 일부를 두 번 이상 읽어야 할 때
2) 문서를 빈번히 수정해야 할 때
3) 문서 구조 처리가 특별히 중요할 때

즉, 문서의 일부분을 정렬하거나 GUI를 통해 사용자가 빈번히 수정해야 할 때가 바로 이런 경우에 해당한다. 하지만 이러 경우에는 XML 문서로부터 반드시 DOM 객체를 생성할 필요가 없다. XML 데이터바인딩으로부터 생성된 객체를 사용하거나 또는 SAX 파서를 통해 문서를 읽으며 그 내용을 다른 객체의 트리구조로 변환하는 경우에는 SAX가 오히려 유리하다.

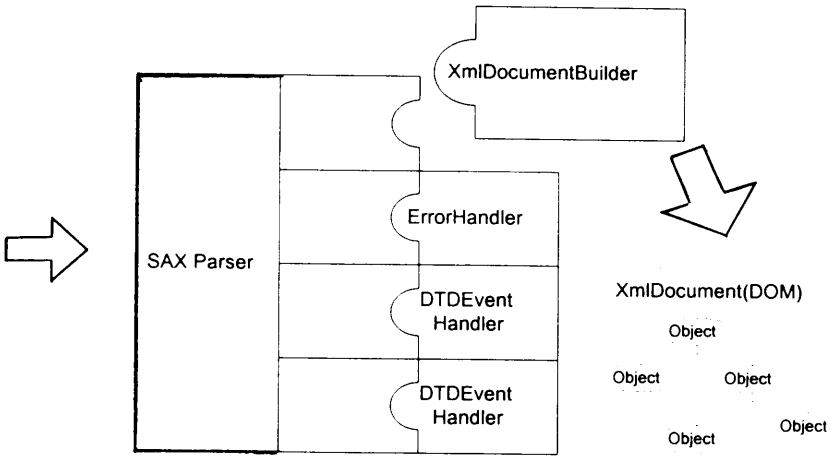

Fig. 1 Structure of DOM parser

실제 DOM 파서들은 대부분 SAX 파서를 내부적으로 사용해 DOM객체를 만들어 낸다. 이런 점을 고려하면 DOM은 다음의 경우에

적합하다.

1) 응용프로그램이 다양한 종류의 XML 문서를 다루는 경우
2) XML 문서의 구조가 충실히 보존돼야 하는 경우

다음은 DOM이 가지고 있는 단점이다.

1) 메모리 사용이 많다. DOM은 문서전체를 메모리에 올려둔다. 따라서 문서가 일정정도 이상 클 경우 메모리 사용량이 지나치게 커질 수 있다. 또 특정 XML 문서의 구조와 무관한 일반적인 구조이므로 실제 처리되지 않는 공백 등도 메모리에 할당한다.

2) 속도가 느리다. DOM 구조를 빈번히 이용하는 경우면 상관없다. 그러나 DOM 구조를 한번 처리하거나 다른 객체의 나무 구조로 변환하고 경우에는 DOM 객체의 생성 비용을 정당화할 수 있다.

3) DOM은 지나치게 일반적이므로 문서의 특성을 충분히 반영하지 못한다.

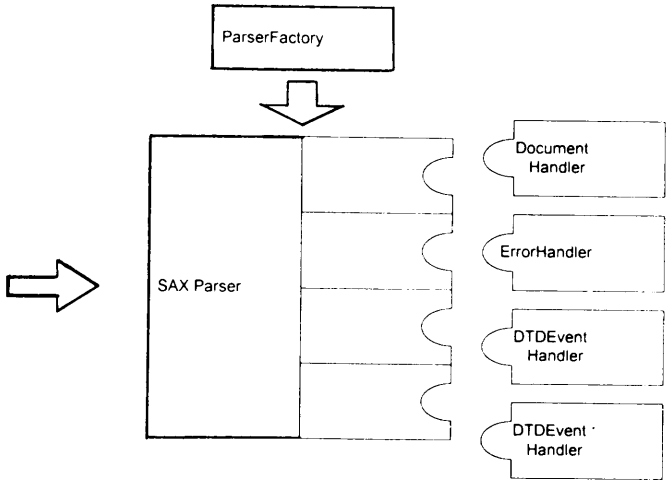

Fig. 2 Structure of SAX parser

DOM에 비해 SAX는 일반적으로 다음과 같은 경우에 적합하다.

1) XML 문서를 순차적으로 일괄 처리하는 경우
2) 상대적으로 XML 문서가 간단하고 그 구조 자체가 중요하지 않은 경우

결국, SAX를 사용하는 것은 DOM과 같은 메모리, 속도와 관련된 문제점을 해결하기 위함이다. 하지만 SAX는 이와 달리 DOM 생성 작업이 단순하지 않다. 실제로도 DOM 파서는 SAX 파서를 이용하되, DOM 생성부분을 추가하고 있는 경우가 대부분이다. SAX는 단순하게 어떤 요소를 읽었는지 등의 정보를 줄뿐, 현재 이 요소가 어떤 요소의 일부인가 등의 문맥정보를 자동으로 유지해주지 않는다. 이는 사용자가 문서 구조로부터 정보를 추출하기 위해 보다 많은 일을 해야 함을 의미한다. 결국 SAX를 사용하는데 있어 성패는 SAX의 단점을 보완할 만한 도구를 적절히 만들 수 있느냐에 달렸다.

본 논문에서는 XML 문서를 순차적으로 읽어 내려가며 각각의 요소들의 연관관계를 데이터베이스에 입력하는 과정을 반복하므로 SAX 파서를 이용하여 구현하였다.

2. XML 문서의 기본

XML 문서는 트리구조를 하고 있다. 하지만 기존의 관계형 데이터베이스는 선형구조를 하고 있다. XML의 내포구조를 선형구조로 나타내기 위해서는 XML 문서를 어떻게 저장할 것인가에 대한 모델설계가 필요하다. 우선 XML 문서를 데이터베이스에 저장하기 위해서는

DTD 문서의 종류에 따라 XML 문서의 유형부터 정의해 볼 수 있다.

1) DTD가 따로 존재하는 문서
2) DTD가 XML문서에 내포되어 있는 문서
3) DTD가 없는 문서

여기서 DTD를 이용해 XML 문서의 유형을 나누기는 하지만, DTD는 문서가 가진 특성을 규정한 문서이기 때문에 어떤 특별한 의미를 가지는 것이라고 보기는 어렵다. 단지 XML 문서가 적합한지만 가려내는 문법이다

하지만 본 논문에서는 XML 문서가 관계형 데이터베이스에 저장이 되었을 경우, XML 문서의 원형 복구뿐만 아니라 DTD의 유형에 의한 문서 생성도 염두해 두고 있기 때문에, DTD 의 저장방식도 유형에 따라 저장방식이 달라진다.

DTD가 따로 존재하는 경우에는 DTD의 이름정보를 저장하게끔 하였고, DTD가 XML 문서 내에 내포되어 있는 경우에는 DTD를 XML 문서 내에서 분리해내어 DTD를 따로 저장하게끔 하였으며, DTD가 없는 경우는 특별한 처리를 해주지 않아도 된다.

DTD가 처리된 후에는 XML 문서 자체에 대한 처리를 해주어야 되는데 XML 문서는 구조화된 문서이기 때문에 몇 가지 패턴이 반복되면서 문서를 이루고 있다. 구조는 크게 PI와 엘리먼트로 나타낸다.

XML 선언부도 특정한 형식의 PI라고 볼 수 있다.

엘리먼트는 크게 태그, 속성, 내용으로 나눌 수 있고, 속성값은 다시 속성과 내용을 나눌 수 있다.

1) 개체정의

XML 문서와 관계형 데이터베이스와 맵핑하는 것은 두 가지 모델이 다른 줄기를 가졌기 때문에 몇 가지 문제가 일어날 수 있다. E.F Codd가 제안한 관계형 데이터베이스 모델은 XML 데이터 모델과 상당히 다르다. 이것은 기본적으로 3계층 모델이다. 데이터베이스는 레코드의 구성으로 된 테이블의 집합이고, 레코드들은 몇 개의 필드로 구성되어 있다. 레코드 필드의 값은 하나여야만 한다. 값들의 리스트이거나 그렇지 않으면 값들의 리스트로 구성된 관계는 될 수 없다.

반면 XML 문서는 불명확한 깊이를 가진 트리구조이다. 그러므로 XML 관점을 관계형 모델로 표현을 해내야 한다. 일반적인 XML 문서를 관계형 구조에 맵핑시킨다는 것은 더욱더 어려운 것이다.

XML 문서가 트리구조임에도 불구하고, XML은 무난한 네임스페이스를 가진다. 엘리먼트타입이 한번이상 선언되는 경우는 없다. 모든 엘리먼트는 같은 영역에 선언되기 때문에 다른 이름을 가져야만 한다. 엘리먼트 타입은 어트리뷰트를 선언할 수 있고 다른 엘리먼트들이 일반적으로 어트리뷰트 네임으로 쓰인다. 이것은 관계형 모델의 제한(restriction)과 비슷하다. 모든 테이블 네임은 다르고 같은 필드이름은 다른 테이블이어야만 한다. 이것은 테이블은 엘리먼트에 필드값에 맵핑시킬 수 있다고 볼 수 있다. 테이블의 각각의 행은 빈 엘리먼트에 일치시키고 어트리뷰트의 값은 필드의 값과 일치시킨다.

다음은 XML 문서를 관계형 데이터베이스에 맵핑하기 위해 몇 가지 요소로 나누어 보았다.

XML 선언부 : XML 문서의 전반적인 정보를 담는 선언부에 대한 개체이다. Encoding정보, 버전(Version), DTD를 구성하는 형태에 대한 정보인 RMD(Required Markup Declaration)를 가진다.

File 요소 : XML 문서가 관계형 데이터베이스에 저장하기 이전 문서를 만들기 위한 Key를 제공하게 된다.

DTD 요소 : XML 문서가 관계형 데이터베이스에 저장된 후에 동일한 DTD에 의한 문서를 추출하기 위해 필요한 테이블이다.

PI 요소 : PI도 일정한 양식의 태그를 가지고 있으므로 Element 속성에 의해서 위치가 결정된다. INDEX값에 의해서 유일해 지며 내용과 처리지시를 따로 저장하게 된다.

Element 요소 : XML 문서는 모두 Element에 의해서 취급되게 된다. File 값은 Element가 속해있던 file의 인덱스를 가리키며, DTD 요소는 DTD의 인덱스를 가리킨다. 다만 DTD가 필요 없는 경우이면, null값을 가지게 된다. 관계형 데이터베이스에서 트리구조를 표현하기 위해서 깊이 값을 넣었다.

속성요소 : Element가 가리키는 것의 속성을 정의 할 수 있다.

속성내용 : 속성값이 가질 수 있는 것은 제한이 없다. 속성값과 같이 묶어버릴 경우, 이 점을 처리하기가 어려워진다. 그래서 따로 테이블을 만들어 부여함으로서, 추가, 삭제, 편집이 보다 쉽게 된다.

내용요소 : 어떤 Element와 관계되어있는지를 나타내는 Element요소와 실질적인 데이터이다. 한 엘리먼트에 속하는 데이터들은 여러개가 있을 수 있다.

2) 테이블 설계 및 관계설정

XML 문서를 저장하기 위해서는 구조적 정보를 데이터베이스에 저장할 수 있어야 하고 저장된 문서에 대해 탐색(Navigation)과 엘리먼트에 대한 질의, 구조적 질의가 가능하게 설계되어야 한다.

트리에 대한 구조 질의를 위해서 LIST를 이용하여 자식 노드를 가리키는 일반적 트리 구성 방식과 특정 트리를 기준으로 구조적 질의를 위한 DFS(Depth First Search)Numbering 방식을 이용하여 트리 구조를 데이터베이스에 함께 표현한다.

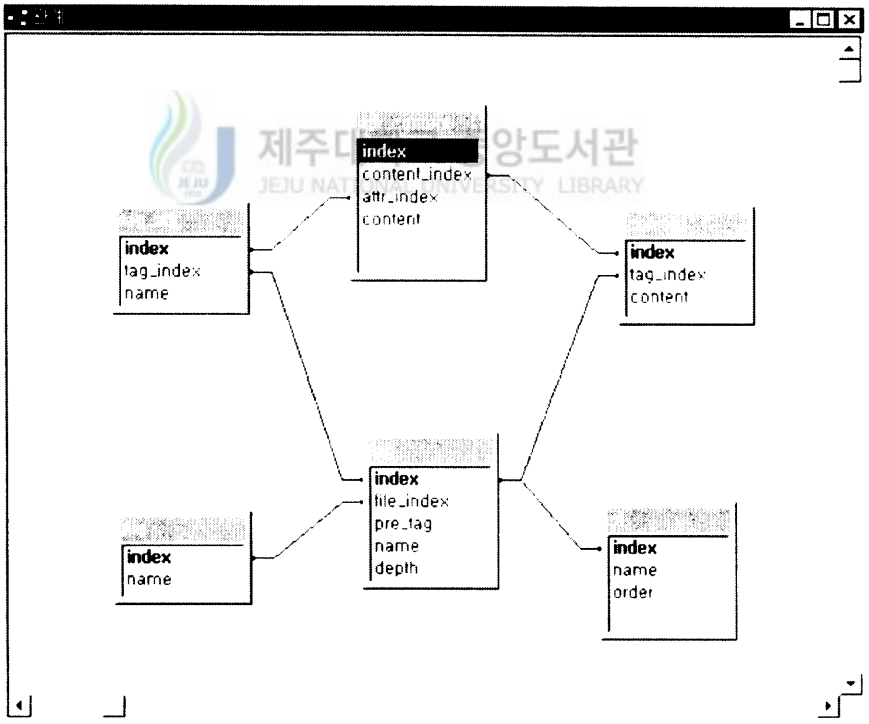

Fig. 3 Relation of table

XML 문서를 데이터베이스에 저장하기 위해서는 먼저 XML 문

시를 데이터베이스에 어떻게 저장하는가에 대한 모델설계가 필요하다. 모델설계는 XML 문서의 기본구조를 살피고 그 기본구조에 따라 XML 문서를 표현하기 위한 개체정의와 개체간의 관계를 규정짓는 과정이 이루어진다.

각각의 내용을 테이블로 만들어 보면 다음과 같다.

file : index, name

tag : index, file_index, pre_tag, name, depth

tag_attr : index, tag_index, name

attr_content : index, content_index, attr_index, content

content : index, tag_index, content

pi : index, name, order

File 테이블

index : 파일을 나타내는 유일한 값

name : 파일 이름

Tag 테이블

index : 동일 DTD에서 유일한 element를 나타내는 값. 같은 name값을 가지고 있지만 depth가 다르면 다른 element이므로 다른 값을 부여받게된다.

file_index : 어떤 XML 파일의 element를 나타내기 위한 값

pre_tag : 바로 이전의 element값

name : element의 이름

depth : 각각의 element들의 깊이 값.

Tag_attr 테이블

index : 인덱스 값

tag_index : 어떠한 element 의 attribute인지 나타내준다.

name : attribute값

Attr_content 테이블

index : 인덱스 값

content_index : content 값마다 다른 attribute를 가질 수 있기 때문에 content에 종속된다.

attr_index : tag_attr테이블과 연결하기 위한 값

content : attribute값

Content 테이블

index : 인덱스 값

tag_index : element의 인덱스

content : 실질적인 내용

Pi 테이블

index : 인덱스 값

name : Pi 값

order : Pi값을 실행시킬 외부 처리기 값

Ⅳ. 시스템 구현

1. 문서저장

XML 문서가 입력으로 들어오면 문서를 한줄씩 읽으면서 테이블을 튜플단위로 분류한다. 먼저 XML 문서 선언부를 읽어서 XML 선언부 테이블에 해당하는 정보를 배열에 저장한다. XML 문서형식 정의부(DTD)가 있는지 확인하고 같이 들어있다면 DTD 부분을 추려낸다. 이를 파일로 만들어서 따로 저장한다.

다음으로 문서부(DI)의 모든 줄을 차례차례 읽는데 이것이 엘리먼트인 경우에는 구조 테이블에 필요한 구조 정보를 구조 테이블 배열에 저장하고, 이 중 구성이 엘리먼트_요소 테이블에 해당하는 정보를 엘리먼트 테이블 배열에 저장한다. 하지만 동일한 DTD나 File이 있을 시에는 동일한 내용이라면 추가를 하지 말아야 할 것이다. 또한 동일한 엘리먼트일 경우 무조건 추가하는 것이 아니라 다른 깊이에 존재할 경우에만 추가한다. 만약에 엘리먼트가 속성들을 가지고 있으면 속성_리스트_값 테이블에 해당하는 정보를 속성_리스트_값 테이블에 저장한다.

SAX파서의 동작에 따라서 문서부(DI)를 모두 읽어서 처리하는 것이 아니라 엘리먼트별로 한줄씩 처리하여 데이터베이스에 맵핑한다

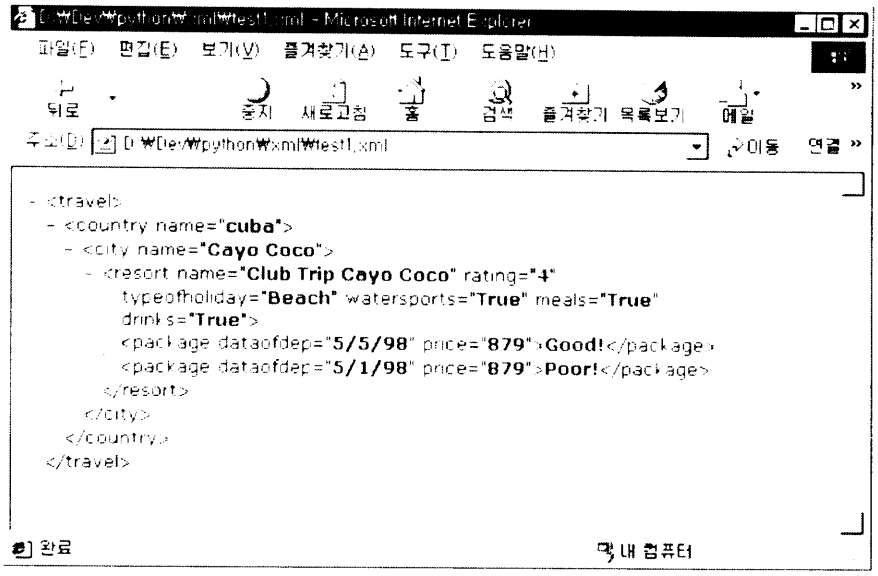

Fig. 4 XML Document

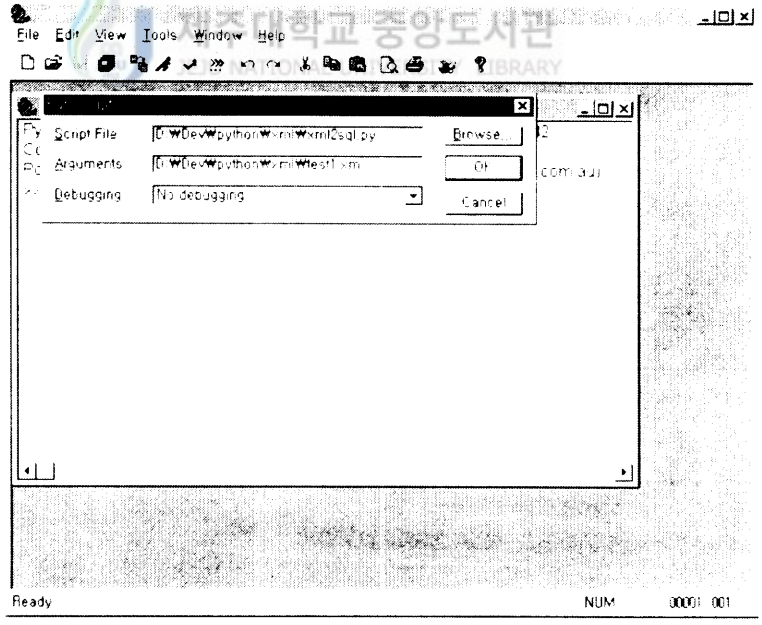

Fig. 5 Execution

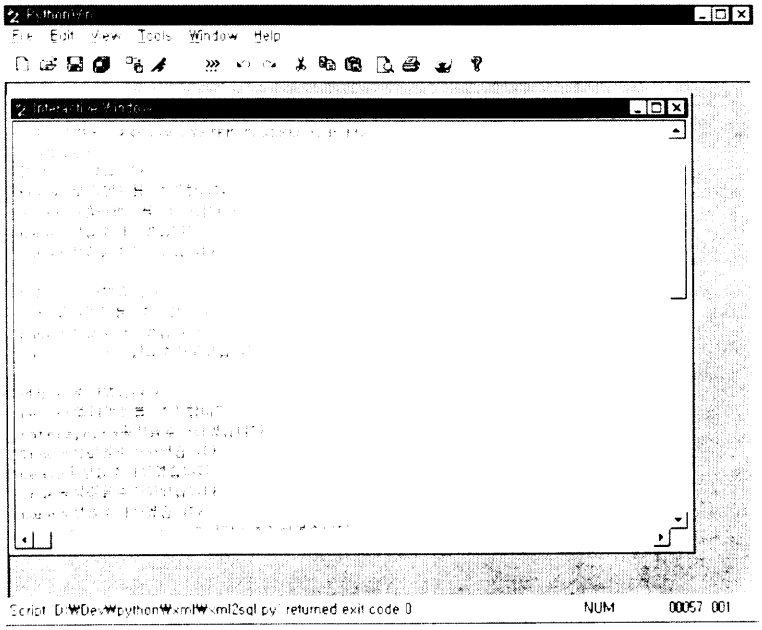

Fig. 6 Result

attr_content : 테이블

index	content_index	attr_index	content
366	0	98	cuba
367	0	99	Cayo Coco
368	0	100	True
369	0	101	True
370	0	102	Club Trip Cayo
371	0	103	True
372	0	104	4
373	0	105	Beach
374	16	106	5/5/98
375	16	107	879
376	17	106	5/1/98
377	17	107	879
(일련 번호)	0	0	

현재 레코드: 1 전체: 12

Fig. 7 Attr_content table

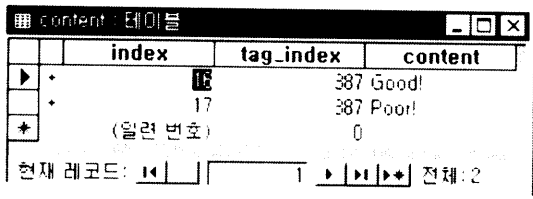

Fig. 8 Content table

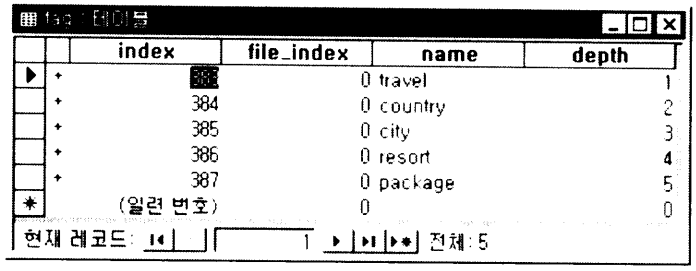

Fig. 9 Tag table

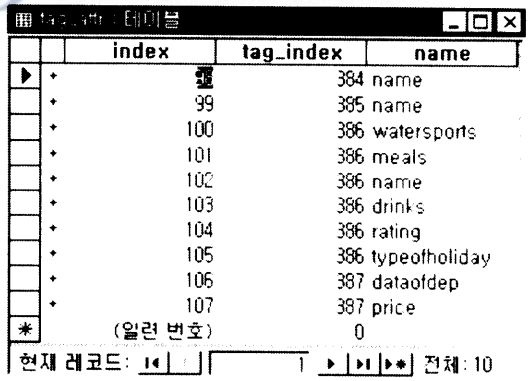

Fig. 10 Tag_attr table

Ⅳ. 결 론

XML은 웹상에서 구조화된 문서를 전송 가능하도록 설계된 표준화된 텍스트 형식이다. 이는 인터넷에서 기존에 사용하던 HTML의 한계를 극복하고 SGML의 복잡함을 해결하는 방안으로 나온 마크업 언어이다.

XML 문서는 이기종 시스템간의 데이터 이동과 같은 미들웨어 등 다양한 응용분야에서 수요가 증가함에 따라 효율적인 저장과 관리가 필요하게 되었다. 이를 위해 XML 문서를 데이터베이스에 저장해야 한다. 본 논문에서는 XML 문서를 구성요소별로 분리하여 관계형 데이터베이스에 저장하는 방안을 제안하였다.

XML은 최근에 개발된 언어로서 XML 문서를 효율적으로 데이터베이스에 저장하는 연구가 아직 부족한 상황이다. 기존 연구들을 살펴보면 관계형 데이터베이스에 XML 문서를 저장하기 위하여 구체적인 모델링 단계와 충분한 검증 없이 테이블을 만들었다. 즉, XML 문서를 데이터베이스에 완전하게 분리 저장하기보다는 대강의 구조적인 정보만을 따로 저장하고 원문에 속하는 엘리먼트의 내용도 따로 저장함에 따라 같은 데이터가 여러 번 반복되어 저장되고 동일한 엘리먼트가 동일한 깊이에 존재하더라도 추가함으로서 저장 면에서 효율이 떨어지며 데이터베이스의 일관성이 깨어진다. 그렇기 때문에 엘리먼트 단위의 수정을 할 수가 없다.

이러한 문제를 해결하기 위해서는 XML 문서를 완전히 분리하여 데이터의 중복이 없게 저장하는 방법이 필요하다. 본 연구에서는 XML 문서를 테이블에 완전히 분리 저장하여 데이터의 중복이 없고, 또한 XML 문서를 다시 재구성할 수 있도록 하기 위해 구체적인 모델

링 단계를 거쳐서 테이블 스키마를 작성하였다. XML 문서를 저장함에 있어 기존연구에서는 내용에 해당하는 부분을 Long Type이나 BLOB으로 저장함에 따라 데이터베이스의 LIKE와 같은 문자 비교를 할 수 없으므로 내용 검색을 하기 위해 별도의 정보검색기를 구축하여야 한다. 그에 반해 본 연구에서는 엘리먼트의 내용부분을 한 줄 단위로 저장함으로 데이터베이스의 LIKE와 같은 문자 비교를 가능하게 하였다. 그러므로 내용 검색을 간단하고 정확하게 할 수 있다.

향후 연구과제로는 XML에 제공하는 링크의 기능과 외부로 보여지기 위한 포맷팅 처리를 지원하는 XML 저장 시스템의 설계와 구현이 필요하다.

Ⅵ. 참고문헌

Extensible Markup Language(XML), "http ://www.w3.org/TR/PR xml 971208"

R. Sack Davis, T. Arnold Moore and J. Zobel, "Database Systems for Structured Documents," Informational Symposium on Advanced Database Technologies and Their Integration, 1994

S. Berchtold. D. A. Keim, H P. Kriegel, The X tree : An Index Structure for High Dimensional Data, Proceeding of the 22nd VLDB Conference

W. Niblack, et. al., "The QBIC project: Quering by Image Content Using Color, Texture, and Shape," Proc. of SPIE Storage and Retrieval for Image and Video Databases, pp. 173 187, 1993

J. R. Smith, S. F. Chang, "VisualSEEk: a Fully Automated Content Based Image Query System," ACM Multimedia Systems, Nov 1996.

Choon Bo Sim, Kwang Tack Song, Jae Woo Chang, Joon Whoan Lee, and Jae Dong Yang, "Design and Implementation of a Content Based Multimedia IR System for Cyber Museums", SPIE Electronic Imaging and Multimedia Systems II, pp 86 93 (1998).[1].

Tim Bray, Jean Paoli, C. M. Sperberg McQueen, Extensible Markup Language (XML) 1.0, REC xml 19980210,

Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, Dan Suciu, XML QL: A Query Language for XML, NOTE xml ql 19980819,W3C,

김용훈, 다양한 구조 검색을 지원하는 XML 문서 검색기의 설계 및 구현, 석사 학위 논문, 충남대학교, 1999.

김용훈, 이강찬, 이규철, 링크 검색을 지원하는 XML문서 질의 언어(XQL)의 설계 Proceedings of The 25th KISS Fall Conference 1998

상현철, "관계 데이터베이스를 이용한 구조화 문서의 색인 및 검색" 충남대학교 석사학위졸업논문, 1998

김현기, 노대식, 강현규, "DocMaster/RM:SGML/XML 문서 관리 시스템" 한국정보처리학회 추계학술발표논문집 제6권 제1호, pp.76 79, 1998
연세원, 조정수, 이강찬, 이규철, "XML 문서 구조검색을 위한 저장 시스템 설계" 정보과학회 학술발표 논문집(B) 26권 1호 , 1999
연세원, 장동준, 김용훈, 이강찬, 이규철, "효율적인 검색지원 SGML 저장관리기의 설계 및 구현" 데이터베이스 학술대회 논문집, 1999
손정한, 한성국, 장재우, 주종철, "SGML 정보 검색 인덱스 설계를 위한 K ary트리, 문서 단위 구문 트리와 엘리먼트 단위 구문 트리의 비교", 98'한국정보과학회 가을 학술 논문집 Vol. 25. No. 2, pp.383 385, 1998

감사의 글

정신없이 살아오면서 뒤를 돌아보는 시간이란 건 정말로 필요하다. 짧은 시간이지만 이런 기회를 빌어서라도 과거를 생각해 볼 수 있다는 것도 다행이다. 하지만 뒤돌아본다고 해서 항상 좋은 일만 있었던 건 아니다. 좋은 일이 있었으면 나쁜 일도 있고, 이런 일상을 반복하면서 지금까지 지내면서 마무리 지은 일이 몇 가지나 있나 한번 생각해보았다. 몇 분을 생각해보아도 생각이 나지 않는건 보면 기억을 못하는 것이 아니라, 하나도 없는 건 아닐까하는 생각이 불현듯 들었다. 이제와서야 부끄럽지만 한가지 생긴 것에 대해 고마움 따름이다. 나 혼자 이일을 해내었다고는 상상도 해보지 못해보았다.

몇 일 되지는 않았지만 타지 생활에서 더욱더 고마움을 느끼고 있고, 지금까지 낳아주시고 키워주시고 뒷바라지 해주신 부모님, 언제나 잔해드리지 못하여 안타깝고 죄송할 따름이다. 부족하지만 언제나 편안하게 대해주시는 곽호영교수님께 부족하지만 이 지면을 빌어서 고마움을 전하고 싶다. 많은 도움을 주신 송왕철교수님께는 더욱 열심히 하는 것으로 고마움을 보답하는 것이라 생각된다. 힘든 부분이 있으면 성성을 다하여 도와주었던 행진선배, 정희선배, 경복선배에게도 고마움 따름이다. 옆에서 헤매고 있을 때 힘이 되어준 성민형, 명휘 모두다 고맙다.