# 객체관계형 데이터베이스에 기반한 XML 문서 저장 및 검색 시스템의 설계 및 구현

이성대* · 곽용원** · 박휴찬***

# Design and Implementation of XML Documents Storage and Retrieval System based on Object-Relational Database

Seong_Dae Lee* · Yong-Won Kwak** · Hyu-Chan Park***

이 논문은 한국해양대학교 BK21사업단의 지원을 받았음.

요 약

XML은 e-비지니스와 많은 응용 프로그램 사이에서 정보를 교환하는 인터넷 표준으로 자리잡고 있다. 따라서 이러한 XML 문서를 효율적으로 관리하기 위해 데이터베이스를 사용하여 저장할 필요성이 있다. 본 논문에서는 객체관계형 데이터베이스를 기반으로 XML 문서를 저장하고 검색하는 시스템을 설계하고 구현하였다. XML 문서를 엘리먼트로 분해하고, 각 엘리먼트 타입에 따라 저장하는 방법을 사용하였다. 또한 저장된 XML 문서를 추출할 수 있는 다양한 형태의 검색을 가능하도록 하였다.

ABSTRACT

XML has emerged as the internet standard for information exchange among e-businesses and applications. Therefore, it becomes necessary to store XML documents in database for efficient management. This paper describes the design and implementation of XML documents storage and retrieval system based on object-relational database. The storage method first, decomposes XML document into element, and then stores according to element types. The system also supports various search methods to retrieve XML documents from database.

키워드

XML, 객체관계형 데이터베이스, 저장 및 검색

## Ⅰ. 서 론

XML(eXtensible Markup Language)은 HTML (HyperText Markup Language) 문서의 구조적 표현에 대한 단점을 보완하고, SGML(Standard Generalized Markup Language)의 복잡성을 간소화하기 위해 W3C(World Wide Web Consortium)에서 웹 문서의 표준으로 지정하였다[1]. XML은 사용자의 확장과 이기종 시스템 간의 정보 교환이 용이하며, 구조적으로 문서를 표현하는 데 유용하다. 이러한 장점으로 인하여 최근 디지털 도서관, 전자상거래, 전자문서교환 등의 다양한 응용 분야에서 활용되고 있다. 이에 따라 XML 문서를 보다 효과적으로 관리하기 위한 연구들이 진행 중이며, 특히 데이터베이스를 이용하여 XML 문서를 저장하고 관리하는 방안에 대한 연구가 활발히 진행되고 있다[2-6].

*한국해양대학교 대학원 컴퓨터공학과
**(주)은성사
***한국해양대학교 기계 · 정보공학부

접수일자 : 2003. 4. 17

XML 문서를 관계형 데이터베이스(Relational Database)에 저장할 경우 검증된 우수한 성능을 쉽게 사용할 수 있는 장점이 있지만 XML 문서의 구조적인 관계를 효과적으로 표현하지 못하고, 집합 값(set-value)을 지원하지 못하는 등 여러 제약들을 가지고 있다[7]. 객체지향형 데이터베이스(Object-Oriented Database)는 객체의 특성을 활용하여 객체들 간의 관계를 효과적으로 저장할 수 있지만 문서의 임의 위치 접근과 검색 시 비효율적인 면을 가지고 있다[8].

본 논문에서는 이러한 문제들을 해결하고, 집합 값과 XML 문서의 구조정보를 효과적으로 표현할 수 있는 객체관계형 데이터베이스(Object-Relational Database)를 기반으로 XML 문서를 저장하고 검색할 수 있는 시스템을 설계하고 구현하였다. 객체관계형 데이터베이스를 사용함으로써 효과적인 저장 스키마 설계가 가능하고, 문서 특성을 유지하면서 정보들을 저장할 수 있으며, 다양한 검색 질의 및 계층적 질의를 빠르게 처리할 수 있다. 본 논문에서는 먼저, XML 문서의 구조정보를 DOM(Document Object Model)을 사용하여 추출한다. 추출된 각 요소들은 특성을 고려하여 전체 문서의 정보를 저장하는 문서 테이블(document table), XML 문서의 구조를 저장하는 구조 테이블(structure table), XML 문서의 각 엘리먼트를 저장하는 엘리먼트 테이블(element table), 각 엘리먼트의 속성들을 저장하는 속성 테이블(attribute table), 내용 테이블(content table)로 구성하였다. 각 테이블은 DTD(Document Type Definition)의 구조정보에 영향을 받지 않도록 문서의 내용을 엘리먼트별로 나누어 저장하는 분할 저장 방식으로 설계하였다. 데이터베이스에 저장된 값들에 대한 다양한 방식의 검색을 위하여 구조 기반 검색, 내용 기반 검색, 혼합 검색과 속성 검색이 가능하도록 하였고, 검색 결과는 XML 문서로 재구성되어 사용자에게 출력되도록 하였다. 그림 1은 본 논문에서 제안한 XML 문서의 저장 및 검색 과정을 나타내고 있다.

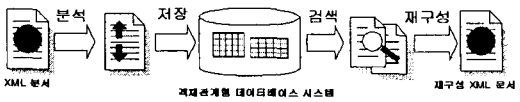



그림 1. XML 문서의 저장 및 검색
Fig. 1 XML Documents Storage and Retrieval

본 논문의 2장에서는 XML 문서의 저장 및 검색 시스템과 관련된 연구들을 살펴보고, 3장에서는 본 논문에서 제안하는 XML 문서를 저장하고 검색하는 시스템을 설계하고, 4장에서는 제안한 방법을 적용하여 구현한 시스템에 대하여 설명하고 기존 시스템과 비교한다. 5장에서는 결론과 함께 향후 연구 과제에 대하여 논의한다.

## II. 관련 연구

### 2.1 XML 문서의 구조 및 처리 기법

XML은 문서의 구조를 정의하는 부분과 문서의 내용을 표현하는 부분으로 나눌 수 있다. 문서의 구조는 일반적으로 DTD를 사용하여 정의하며, DTD는 엘리먼트, 엘리먼트들의 속성을 나타내는 속성 목록(attribute list), 그리고 공통으로 사용되는 엔티티(entity)로 구성된다[9]. XML 문서의 내용은 DTD에서 정의한 엘리먼트와 엘리먼트의 속성들을 사용하여 표현한다.

XML 문서는 XML 선언부, DTD 연결부, 그리고 DTD에 따른 실제 문서 내용으로 구성된다. XML 선언부에는 이 문서가 XML 문서라는 것과 이 문서의 XML 표준 버전을 선언한다. DTD 연결부에서는 XML과 DTD를 연결하는 방법을 명시하며, XML 문서에 DTD 정의 부분을 직접 포함시키는 방법과 외부에 선언된 DTD 파일의 위치를 지정하여 연결하는 방법이 있다. 또한 DTD가 없는 XML 문서도 존재할 수 있다. 마지막으로 XML 문서의 실제 내용은 정의한 DTD에 따라 작성되어야 한다.

```
<!-- DTD for BankInfo -->
<!ELEMENT BankInfo   (Customer*)>
<!ELEMENT Customer   (Name, Address, Account+)>
<!ATTLIST Customer   CustomerID ID #REQUIRED>
<!ELEMENT Name       (#PCDATA)>
<!ELEMENT Address    (City, Street, Code)>
<!ELEMENT City       (#PCDATA)>
<!ELEMENT Street     (#PCDATA)>
<!ELEMENT Code       (#PCDATA)>
<!ELEMENT Account    (SaveAmount*)>
<!ATTLIST Account    AccountNO NO #REQUIRED>
<!ELEMENT SaveAmount (Won*)>
<!ELEMENT Won        (#PCDATA)>
```

그림 2. 은행고객 DTD
Fig. 2 DTD for Bank Customer

```
<? xml version = "1.0"?>
<! DOCTYPE BankInfo SYSTEM "bank.dtd">
<BankInfo>
  <Customer CustomerID = "1234">
    <Name>Hong Gil-Dong</Name>
    <Address>
      <City>Busan</City>
      <Street>YoungDo</Street>
      <Code>12-34</Code>
    </Address>
    <Account AccountNO = "123-45-56778">
      <SaveAmount>
        <Won>50,000,000</Won>
      </SaveAmount>
    </Account>
  </Customer>
  <Customer CustomerID = "3423">
    <Name>Lee Sun-Sin</Name>
    <Address>
      <City>Busan</City>
      <Street>Haeundae</Street>
      <Code>432-34</Code>
    </Address>
    <Account AccountNO = "123-45-66666">
    </Account>
  </Customer>
</BankInfo>
```

그림 3. 은행고객 XML 문서
Fig. 3 XML Document for Bank Customer

그림 2와 3은 문서 구조 정의 부분인 DTD와 이 DTD에서 명시한 규칙에 따라 작성한 XML 문서의 예제를 보여주고 있다.

XML 문서를 처리하기 위해서는 XML 파서(parser)가 사용되어진다. DOM은 XML 파서에 내장되어 XML 문서의 정보를 추출하거나 수정, 삭제, 갱신 등의 처리 기능을 제공하고 있으며, XML 문서 처리에 사용되는 함수와 변수, 상수들을 정의하고 있는 표준이다[9]. DOM에서는 XML 문서를 트리로 표현한다. 그림 4는 DOM을 사용하여 XML 문서를 트리로 변환한 예제이다.

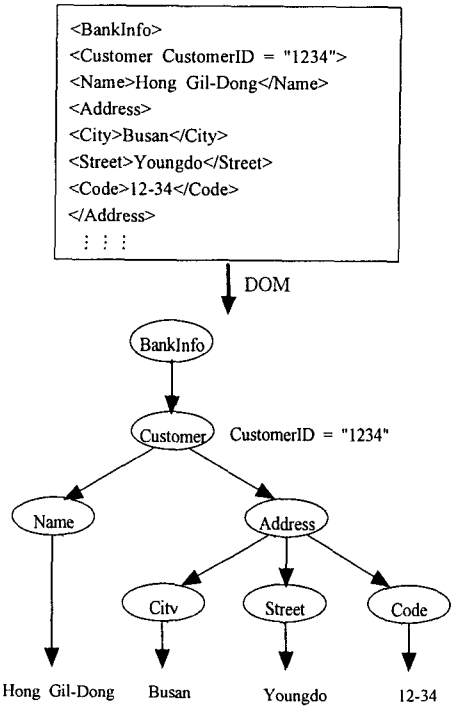



그림 4. XML 문서의 DOM 트리
Fig. 4 DOM Tree of XML Document

## 2.2 XML 문서의 저장 기법

### 2.2.1 저장 방식에 따른 분류

XML 문서의 정보를 데이터베이스에 저장하는 방법으로는 XML 문서 데이터들을 처리하는 방식에 따라 분할 저장 방식과 가상 분할 저장 방식으로 나누어진다[4].

분할 저장 방식은 XML 문서를 DTD의 엘리먼트 단위로 나누어 저장하고, 검색시 구조정보를 참조하여 해당 엘리먼트나 하위 엘리먼트를 조합하여 처리하는 방식이다. 이 방식은 문서의 편집과 관리가 쉬우며 동일한 내용을 가지는 노드를 공유할 수 있다는 장점이 있다. 그러나 문서의 검색에서는 검색 결과를 재구성하는 과정이 복잡하고 시간이 많이 소요되는 단점이 있다.

가상 분할 저장 방식은 하나의 BLOB(Binary Large Object) 형태의 필드에 문서 전체를 저장하고, 실제 문서에서 각 엘리먼트의 위치 정보를 추가적으로 저장하는 방식이다. 따라서 검색 결과의 재구성이

필요하지 않으며, 단말 노드를 검색할 경우 엘리먼트의 위치 정보를 이용한 구조적 검색이 가능하다. 그러나 XML 문서의 일부분에 대한 수정이 발생하면 문서 내의 모든 엘리먼트들의 위치 정보도 수정해야 하므로 이에 따른 과부하와 데이터베이스의 일관성 유지에 문제가 발생할 수 있다[10].

2.2.2 DTD 처리 방식에 따른 분류

DTD는 XML 문서의 구조를 기술하는 표준화된 방법이며, XML 문서에 적용하는 마크업 선언들의 집합으로 구성된다[9]. 이런 DTD를 다루는 방식에 따라 DTD에 종속적인 기법과 독립적인 기법으로 분류할 수 있다.

DTD 종속적 기법은 XML 문서의 구조를 정의하고 있는 각각의 DTD에 맞게 저장 스키마를 생성하는 것이다. 이 방법은 각각의 DTD에 적합한 스키마를 생성하기 위해 스키마 생성기가 필요하다[11]. 또한 XML 문서를 정보의 손실 없이 가장 효과적이고 정확히 저장할 수 있는 장점이 있다. 그러나 DTD의 내용에 변화가 발생할 경우 전체 XML 문서의 구조 정보를 재구성해야 하며, DTD의 수가 늘어나면 저장 모델의 수가 증가하는 단점이 있다.

DTD 독립적 기법은 모든 DTD를 만족시킬 수 있는 하나의 스키마 모델을 구성하여 저장하는 방식이다[3]. 따라서 모든 XML 문서를 DTD에 영향을 받지 않고 저장할 수 있으므로 하나의 저장 모델만 구성하면 모든 문서를 저장할 수 있다. 또한 DTD의 변경으로 XML 문서의 구조정보를 수정할 경우 해당 부분의 정보만 수정해 주면되므로 수정과 갱신이 용이하다. 하지만 문서를 검색할 경우 문서의 재구성 과정이 복잡하고, 각 문서의 구조를 고려하여 저장할 수 없다는 단점이 있다.

2.2.3 데이터베이스에 따른 기법

현재 상용화된 데이터베이스 관리시스템으로는 크게 관계형 데이터베이스, 객체관계형 데이터베이스, 객체지향형 데이터베이스로 나눌 수 있으며, XML 문서의 저장을 위해 각 데이터베이스를 이용할 경우 아래와 같은 특징이 있다.

관계형 데이터베이스를 위한 데이터 모델링은 XML 문서를 구성하고 있는 객체들 간의 관계를 먼저 구성하고, 이 정보들을 관계형 데이터베이스의 테이블의 형태로 구성하여 저장한다[5 8]. 관계형 데이터베이스를 사용함으로써 기존의 많은 사용자들에 의해 사용자 확보와 관리가 쉽다는 장점이 있다. 하지만 관계형 데이터베이스를 사용할 경우 XML과 같은 구조적인 데이터의 정확한 표현이 어렵고, 검색에서 다수의 테이블에 대한 조인 연산으로 시스템의 성능이 저하되고, 집합 값을 지원하지 않는다는 단점이 있다.

객체지향형 데이터베이스를 이용하여 저장할 경우 XML 문서에 표현되는 엘리먼트의 전후 종속 관계는 클래스를 기반으로 하여 객체들 간의 링크로 표현할 수 있어 효율적이다[13]. 하지만 문서 내에 임의의 엘리먼트에 접근할 수 있는 방법이 없기 때문에 객체 검색 시 모든 객체들을 검색해야하는 단점이 있으며, 현재 이런 문제점을 극복하기 위한 연구가 진행 중이다.

최근에는 XML 문서의 저장 효율성을 위해 관계형 및 객체지향형 데이터베이스의 장점을 수용한 객체관계형 데이터베이스를 이용하는 방법이 연구되고 있다[3,12]. 객체관계형 데이터 모델(Object-Relational Data Model)은 객체지향성을 지원하고 다양한 데이터 형을 제공하도록 관계형 데이터 모델을 확장하였고, 추가된 데이터 형을 처리하기 위해 SQL과 같은 관계형 질의어를 확장하였다[14]. 또한 각 튜플들의 속성이 복합 객체형으로 사용되어질 수 있다. 이와 같은 확장은 데이터에 대한 모델링 기능을 확장하면서 데이터로의 선언적인 접근 방식과 같은 관계형의 기초를 유지하며 이루어진다. 그러므로 객체관계형 데이터베이스는 객체지향 특성을 사용하고자 하는 관계형 데이터베이스 사용자에게 편리한 환경을 제공하고 있다. XML 문서의 저장을 위해 이러한 객체관계형 데이터베이스를 이용할 경우 객체지향형 데이터베이스에서 지원하는 복합 객체 타입을 지원할 수 있으며 사용자에 의한 정의 타입과 함수를 구현할 수 있고, 집합 값의 저장도 가능하게 된다. 또한 엘리먼트의 구조정보 표현을 위한 방법으로 객체 ID를 이용해 활용할 수 있으며 상속성과 외부 언어와의 통합 기능을 활용하여 저장 시스템을 설계할 수 있다[3]. 따라서 XML의 구조적인 정보를 보다 효과적으로 표현할 수 있으며 문서 정보 질의 시 계층에

대한 질의 조인을 이용하는 것보다 빠르게 처리할 수 있고 대용량 데이터에 대한 질의를 보다 안정성 있게 처리할 수 있다.

## III. 시스템 설계

### 3.1 데이터베이스 설계

본 논문에서는 XML 문서의 저장과 검색의 효율성을 고려하여 관계형 및 객체지향형 데이터베이스의 장점을 이용할 수 있는 객체관계형 데이터베이스를 사용하여 시스템을 구성하였다. 저장 스키마는 문서의 저장 및 수정이 용이하도록 DTD에 독립적인 모델로 설계하였으며, 각 객체를 분할하여 처리할 수 있도록 분할 저장 방식을 사용하여 XML 문서의 구성 요소의 특성을 고려한 스키마 설계가 가능하도록 하였다.

DTD에 독립적인 스키마를 설계하기 위해서는 모든 XML 문서를 만족시킬 수 있는 저장 스키마를 설계하여야 한다. 이러한 저장 스키마는 XML 문서를 구성하고 있는 요소들의 특성을 파악하여 분류함으로써 설계할 수 있다.

본 논문에서는 XML 문서의 특성을 다섯 개로 분류하여 테이블들을 설계하였다.

1) XML 문서 테이블(XML_Doc)

XML 문서 테이블은 각 문서의 정보를 저장한다. 문서 ID에는 각 문서의 고유 ID를 저장하고, 해당 문서의 이름은 Doc_name에 저장한다. 그리고 문서의 구조를 정의하고 있는 DTD의 이름을 저장하고 DTD_File은 해당 DTD의 전체 내용을 저장한다.

| Doc_ID | Doc_Name | DTD_Name | DTD_File |
|---|---|---|---|
| 문서 ID | 문서 이름 | DTD 이름 | DTD 파일 |

2) XML 구조 테이블(XML_Str)

XML 구조 테이블은 DOM 트리의 루트 노드로부터 깊이우선탐색(depth first search)을 통하여 부여한 각 노드의 ID를 엘리먼트의 순서 정보로 나타내어 저장한다. 또한 소속된 문서의 ID와 엘리먼트의 속성을 저장하고 있는 속성 테이블(XML_Att)의 속성 ID, 엘리먼트의 내용을 저장하고 있는 내용 테이블(XML_Con)의 내용 ID를 추가적으로 저장한다.

| Doc_ID | Ele_ID | Ele_Name | Con_ID | Att_ID |
|---|---|---|---|---|
| 문서 ID | 엘리먼트 ID | 엘리먼트 이름 | 엘리먼트 내용 ID | 엘리먼트 속성 ID |

3) XML 엘리먼트 테이블(XML_Ele)

엘리먼트 테이블은 XML 문서의 구조를 파악할 수 있는 데이터들로 구성되어 있다. 소속된XML 문서의 ID, 엘리먼트 ID, 해당 엘리먼트의 자식 엘리먼트와 형제 엘리먼트 그리고 엘리먼트의 경로 정보를 저장한다. 이렇게 함으로써 엘리먼트들과의 관계와 전체적인 구조를 유지하면서 데이터들을 처리할 수 있게 된다.

| Doc_ID | Ele_ID | Child_Ele | Sibling_Ele | PathExp |
|---|---|---|---|---|
| 문서 ID | 엘리먼트 ID | 자식 엘리먼트 | 형제 엘리먼트 | 엘리먼트의 경로 정보 |

4) XML 속성 테이블(XML_Att)

속성 테이블은 각 속성을 구분하는 속성 ID와 해당 속성을 포함하는 엘리먼트 ID, 속성의 이름과 값을 저장하게 된다.

| Doc_ID | Att_ID | Ele_ID | Att_Name | Att_Value |
|---|---|---|---|---|
| 문서 ID | 속성 ID | 소속 엘리먼트 ID | 속성 이름 | 속성 값 |

5) XML 내용 테이블(XML_Con)

내용 테이블은 해당 내용을 포함하고 있는 엘리먼트 ID, 실제 값을 저장한다.

| Doc_ID | Ele_ID | Con_ID | Con_Value |
|---|---|---|---|
| 문서 ID | 소속 엘리먼트 ID | 내용 ID | 내용 값 |

그림 5는 위에서 정의한 XML 문서를 분류하여 저장하는 각 테이블의 관계도이다.

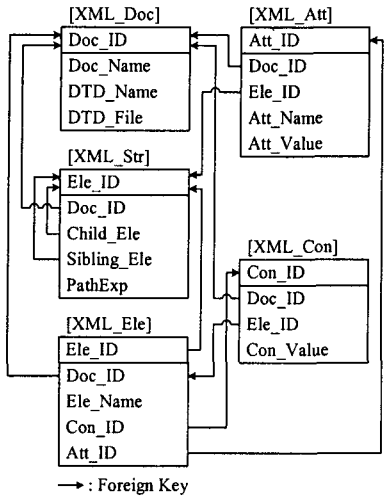



그림 5. 테이블 관계도
Fig. 5 Relationship among Tables

### 3.2 XML 문서 분석 및 저장

사용자에 의해 XML 문서를 읽게 되면 XML 파서에 의해 문서의 구문을 분석하고 유효성 여부를 검사하여 DOM 객체 트리를 생성한다. 이렇게 생성된 DOM 객체 트리에 의해 각각의 요소별로 정보를 추출하기 위해 노드들을 순회하게 된다.

본 논문에서는 트리 노드에서 정보를 추출하여 객체들 간의 관계를 나타내기 위해 엘리먼트에 ID 부여하며 XML 문서의 구조정보를 유지하기 위한 방법으로 깊이우선탐색 기법을 사용하고 있다[2]. 깊이우선탐색 기법은 트리의 루트 노드로부터 깊이우선탐색 기법으로 노드를 방문하여 처음 방문할 때의 순서와 자신의 자식 노드 방문이 모두 끝난 뒤 노드의 방문 순서를 구성하는 방식이다.

그림 6은 XML 문서를 루트 노드로부터 깊이우선탐색 기법을 이용하여 순회하면서 객체에 대한 ID를 부여하고 XML 문서에 대한 구조정보를 나타내고 있다.

XML 문서를 순회하는 동안 각 노드들의 정보는 해당 요소별로 구분하여 스키마 테이블의 구조에 맞게 분류한다. 노드가 엘리먼트이면 순회시 부여한 객체 ID를 엘리먼트의 ID로 설정하여 엘리먼트 테이블에 저장한다. 속성 유무를 체크하여 속성은 속성 테이블로, 내용은 내용 테이블에 각각 관련 정보들을 분류하고 분류가 완료되면 해당 테이블에 저장하게 된다.

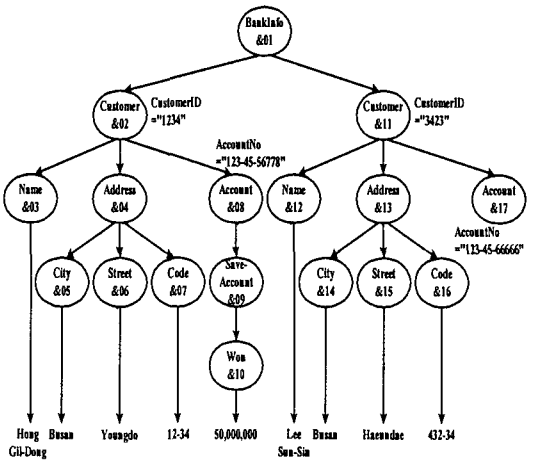



그림 6. 깊이우선탐색을 이용한 ID부여
Fig. 6 ID Numbering by Depth First Search

## IV. 시스템 구현

### 4.1 시스템 구성에 따른 문서 저장 및 검색

본 논문에서 구현한 XML 문서의 저장 및 검색 시스템은 객체관계형 데이터베이스 시스템인 Oracle 8i상에서 구현하였다.

그림 7은 본 논문에서 제안한 방법을 구현한 시스템의 전체적인 구성도로서, 다섯 개의 모듈로 구성되어 있다. 먼저 XML 문서를 저장할 경우, XML 문서를 읽어 유효성을 검사하고, DOM 객체 트리를 생성한다. 저장 관리기는 생성된 DOM 객체 트리를 설계된 스키마별로 분류하여 저장에 필요한 정보들을 관리한다. 인스턴스 관리기는 저장 관리기로부터 분류된 정보들을 데이터베이스에 저장한다. 질의 처리기에서는 사용자의 질의를 입력받아 질의를 수행하고 결과 값이 반환되면 XML 문서 생성기에 값들을 전달하게 된다. 이 결과 값을 가지고 XML 문서 변환기에서는 XML 문서로 재구성하여 사용자에게 보여주게 된다.

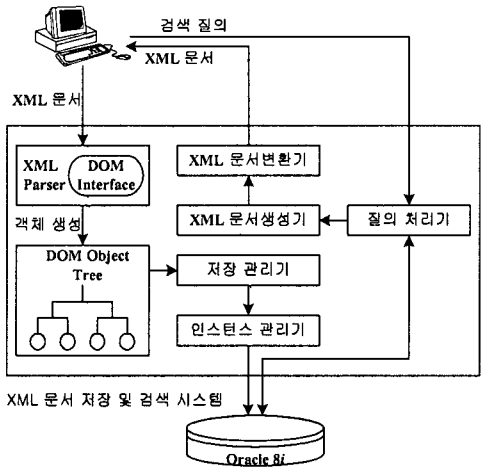



그림 7. XML 문서 저장시스템의 구성
Fig. 7 XML Document Storage System

그림 8은 본 논문에서 제안한 시스템의 흐름도이다. XML 문서를 저장하는 과정을 보면, 먼저 XML 파서는 XML 문서의 유효성을 검사하고 내장된 DOM 인터페이스를 사용하여 XML 문서를 DOM 객체 트리로 변환한다. 이렇게 변환된 DOM 객체 트리가 유효한 정보들이면 저장 관리기에서는 DOM 트리의 객체 정보를 DTD 독립적인 스키마 구조에 적합하도록 각 엘리먼트에 객체 ID를 부여하고, 엘리먼트, 속성, 내용 등으로 분류한다. 또한 XML 문서 저장을 위한 전반적인 데이터 및 정보들도 관리한다. 데이터베이스에 대한 저장 명령이 실행되면 인스턴스 관리기에서는 저장 관리기에서 저장에 필요한 각각의 데이터들을 입력받아 생성한 저장 스키마의 각 요소들과 일치시켜 테이블에 값들을 저장하게 된다.

XML 문서가 저장된 데이터베이스에서 정보를 검색하는 방법에는 크게 구조 기반 검색과 내용 기반 검색이 있다[15]. 본 논문에서는 검색 인터페이스를 통하여 다양한 질의 요청에 대한 처리를 위해 구조 기반, 내용 기반, 구조·내용 기반 검색 방식과 속성을 이용한 검색도 가능하도록 설계하였다.

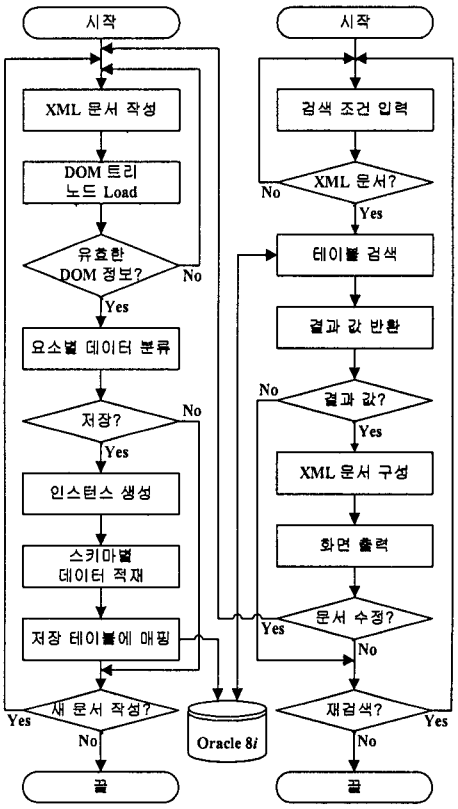



그림 8. XML 문서 저장 및 검색 흐름도
Fig. 8 Flowchart for XML Documents Storage and Retrieval

그림 9는 BankInfo.xml 문서가 저장된 데이터베이스에 대한 다양한 검색 요청에 대한 검색 질의 예를 보여주고 있다. 질의 조건에 대한 검색 결과는 데이터베이스에서 반환된 값을 재구성하여 XML 문서 형식으로 사용자에게 보여주게 된다. 검색 결과에 대한 데이터는 각 요소별로 문서 데이터와 구조 데이터, 속성 데이터, 엘리먼트 데이터, 내용 데이터를 리턴 받게 되는데 이 데이터들을 문자열 배열을 이용하여 재구성한다.

```
1. 구조 기반 검색
  // 'Account' 엘리먼트의 자식노드로
  // 'SaveAccount' 엘리먼트가 있는 문서를 검색
SELECT deref(Doc_ID).Doc_Name
FROM XML_Str WHERE PathExp LIKE
'%Account/SaveAccount%'

2. 내용 기반 검색
  // 'Youngdo'이라는 단어를 포함하는 문서를 검색
SELECT deref(DOC_ID).Doc_Name
FROM XML_Con WHERE Con_Value LIKE
'%Youngdo%'

3. 구조·내용 기반 검색
  // 'Address'의 자식노드로 'Street'를 포함하고
  // 그 값이 'Youngdo'인 문서를 검색
SELECT deref(Doc_ID).Doc_Name
FROM XML_Str
WHERE PathExp LIKE '%Address/Street%'
AND deref(deref(Ele_ID).Con_Value).Con_Value
= 'Youngdo'

4. 속성 기반 검색
  // 속성 'CustomerID'가 '3423' 인
  // 엘리먼트를 검색
SELECT deref(Ele_ID).Ele_Name
FROM XML_Att
WHERE Att_Name = 'CustomerID'
AND Att_Value='3423'
```

그림 9. XML 문서 검색 예
Fig. 9 An Example of XML Document Retrieval

그림 10은 XML 문서를 재구성하는 과정을 나타내고 있다. 문서 재구성은 첫 번째로 반환된 구조정보의 데이터를 배열에 차례로 입력하여 엘리먼트를 우선 생성한다. 이렇게 문서 구조가 구성되고 나면 두 번째 단계로 속성 값을 가져와서 순서에 맞추어 해당 엘리먼트에 삽입한다. 마지막으로 엘리먼트의 내용을 가져와서 해당 엘리먼트의 시작태그 뒤에 값들을 삽입하게 된다. 이와 같이 생성된 XML 문서는 사용자 인터페이스를 통해 사용자에게 출력된다.

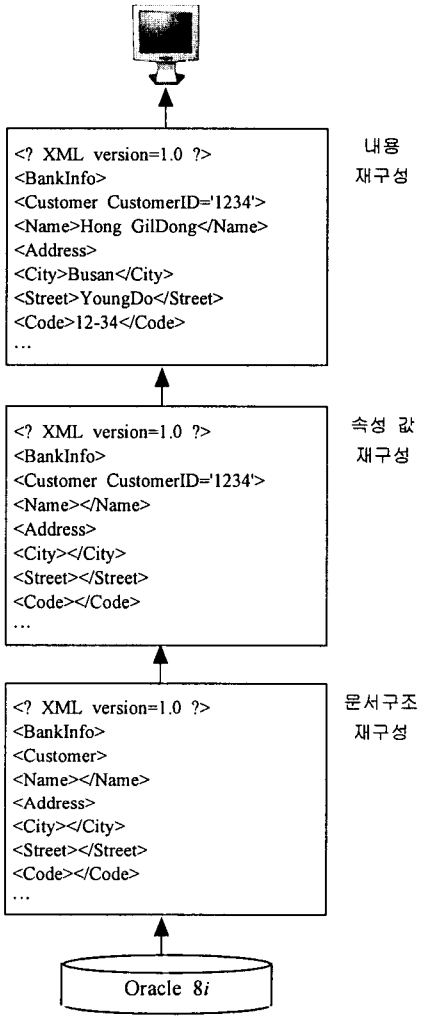



그림 10. XML 문서 재구성
Fig. 10 XML Document Reconstruction

## 4.2 XML 문서 저장 및 검색 시스템 구현

본 시스템은 Windows 2000 Sever 환경에서 데이터베이스로 Oracle 8i, 개발 도구로는 Visual Basic 6.0을 사용하였다. XML 문서의 처리를 위해 DOM이 내장된 Microsoft사의 MSXML 3.0 Parser를 사용하였다.

그림 11은 예제 XML 문서 BankInfo.xml의 DOM 객체의 트리 구조를 나타내며, 그림 12는 BankInfo.xml 문서를 객체관계형 데이터베이스에 저장하기 위해 DOM 객체 트리에서 각 노드들을 순회하며 각 스키마별로 요소들을 분류한 내용과 XML 문서의 내용이다.

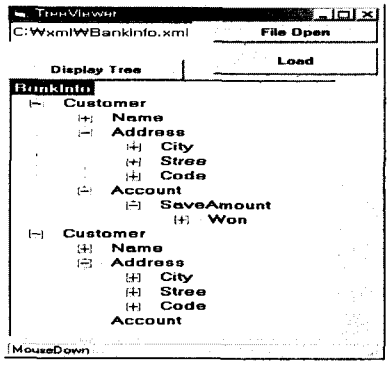



그림 11. XML 문서 트리
Fig. 11 XML Document Tree

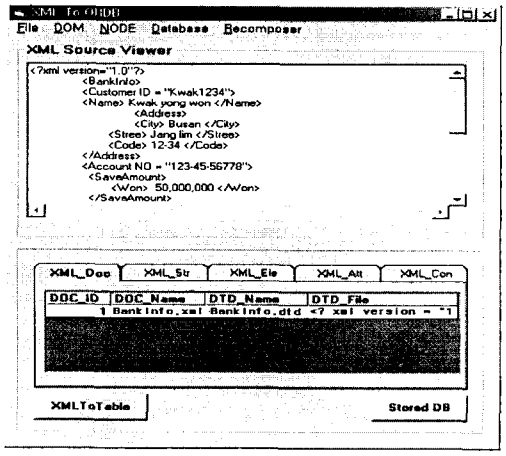



그림 12. XML 저장 인터페이스
Fig. 12 XML Storage Interface

그림 13은 질의 인터페이스에서 입력받은 질의를 수행하고 그 결과 값을 XML 문서 생성기에 의해 해당 XML 문서 구조에 맞게 재구성한 내용을 보여주고 있다.

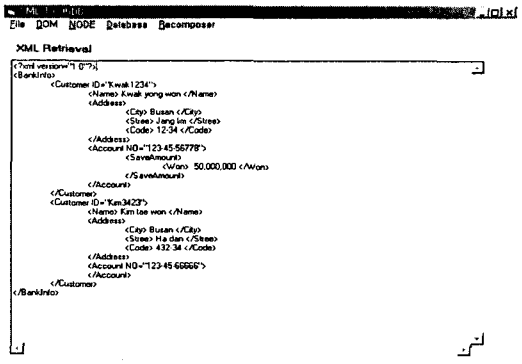



그림 13. XML 문서 재생성
Fig. 13 XML Document Retrieval

### 4.3 기존 시스템과의 비교

본 논문에서 제안한 시스템은 객체관계형 데이터베이스를 이용하여 XML 문서를 보다 효율적으로 저장과 검색을 수행할 수 있도록 하였다. [2]에서 제안한 XML 문서 저장시스템은 데이터 추출 방법이나 스키마 생성 방법, 스키마 매핑 등을 효과적으로 처리할 수 있도록 설계하였지만, 관계형 데이터베이스를 저장 시스템으로 사용함으로써 XML 문서 구조에 대한 충분한 정보를 저장할 수 없고, 구조정보를 표현하기 위해 상대적으로 많은 테이블과 튜플들이 필요하게 된다. 이로 인해 검색에서 여러 테이블에 대하여 조인 연산이 이루어지므로 검색을 수행하는데 보다 많은 시간과 노력이 필요하다.

표 1은 각 데이터베이스에 기반하여 구현한 XML 문서 관리 시스템들을 비교하고 있다.

표 1. XML 저장 시스템 비교
Table. 1 Comparison of XML Storage Systems

| 저장시스템 / 항목 | RDB에 기반한 시스템 [6, 8] | OODB에 기반한 시스템 [13] | ORDB에 기반한 시스템 [본 논문] |
|---|---|---|---|
| 구조정보 표현 | 낮음 | 높음 | 높음 |
| 복합질의 처리 | 높음 | 낮음 | 높음 |
| 검색 속도 | 낮음 | 높음 | 높음 |
| 집합 값 | 비지원 | 지원 | 지원 |
| 사용자 수 | 많음 | 적음 | 많음 |

## V. 결론 및 향후 연구 과제

본 논문에서는 관계형 데이터베이스 시스템과 객체지향형 데이터베이스의 장점을 수용한 객체관계형 데이터베이스 시스템을 이용하여 XML 문서를 저장하고 검색할 수 있는 시스템을 설계하고 구현하였다.

XML 문서를 보다 효과적으로 저장하기 위해 XML 문서를 엘리먼트 단위로 나누어 저장하는 분할 저장 방식을 사용하였으며, DTD에 관계없이 XML 문서를 저장할 수 있도록 DTD에 독립적인 모델을 설계하였다. 그리고 다양한 검색 질의를 지원하기 위하여 내용 기반, 구조 기반, 내용·구조 기반, 속성에 대한 질의를 수행할 수 있도록 질의 처리 시스템을 개발하였다. 본 시스템은 크게 세 부분으로 나누어 개발하였다. XML 문서를 입력받아 데이터를 추출하는 데이터 관리 부분, 추출된 데이터들을 데이터베이스에 저장하기 위한 저장 관리 부분, 질의 수행과 검색된 결과 값을 재구성하는 질의 처리를 담당하는 부분으로 개발하였다.

향후 연구 과제로는 보다 편리한 사용자 인터페이스를 제공할 수 있도록 연구하고, 웹상의 클라이언트/서버 시스템에 접목시켜 인터넷상에서 XML 문서를 저장하고 관리할 수 있는 시스템에 대한 연구가 필요하다.

## 참고문헌


[1] W3C, eXtensible Markup Language (XML) 1.0, http://www.w3.org/1998/REC- xml19980210.html, Feb. 1998.

[2] 이용석, 손기락, "XML 문서 저장 시스템 설계 및 구현", 한국정보과학회 학술 발표 논문집(Ⅰ), 25권 2호, 1998.

[3] T. Shimura, M. Yoshikawa, and S. Uemura, "Storage and Retrieval of XML Documents using Object-Relational Databases", Database and Expert Systems Applications, 1999.

[4] 김훈, 한상웅, 홍의경, "XML 문서 저장 시스템", 데이터베이스 연구회지, 16권 2호, 2000.

[5] R. Bourret, C. Bornhovd, and A. Buchmann, "A Generic Load/Extract Utility for Data Transfer Between XML Document and Relational Databases", Workshop on Advanced Issues of E-Commerce and Web/based Information Systems 2000, pp.134-143, 2000.

[6] A. Schmidt, M. Kersten, M, Windhouwer, and F. Wass, "Efficient Relational Storage and Retrieval of XML Documents", WebDB(Informal Proceedings), 2000.

[7] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, and J. Naughton, "Relational Databases for Querying XML Document: Limitations and Opportunities", Proc. of the 25th VLDL Conf. Edinburgh, Scotland, pp.302-314, 1999.

[8] D. Florescu and D. Kossmann, "Storing and Querying XML Data Using and RDBMS", Data Engineering, bulletin, vol.22, no.3, pp.27-34, 1999.

[9] J. Britt, T. Duynstee, and V. Honnay, Professional Visual Basic 6 XML, WROX Press, 2000.

[10] S. Malaika, "Using XML in Relational Database Applications", Data Engineering, 1999.

[11] 고승규, 조승기, 백승욱, 이경호, 최윤철, "SGML 문서 검색시스템의 설계 및 구현", 정보 과학회, 99 가을 학술발표논문집, 제 26권 1호, 1999.

[12] B. Surjanto, N. Ritter, and H. Loeser, "XML Content Management based on Object-Relational Database Technology", Web Information Systems Engineering, 2000.

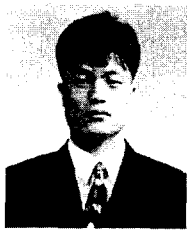
[13] 김영일, 신동욱, 권택근, 김형선, "객체지향형 데이터베이스를 이용한 XML 문서 저장 시스템 설계", 정보 처리 학회지, 1999.

[14] A. Silberschatz, H. F. Korth, and S. Sudarshan, Database System Concepts, 3/E, McGraw-Hill, 1997.

[15] J. Shanmugasundaram, E. Shekita, J. Kiernan, R. Krishnamurthy, E. Viglas, J. Naughton, and I. Tatarinov, "A General Technique for Querying XML Documents using a Relational Database System", Special Interest Group on Management of Data Record, 2001.

[16] M. Graves, Designing XML Databases, Prentice-Hall, 2001.

## 저자소개

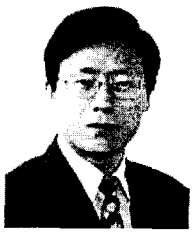


이성대(Seong-Dae Lee)

1999년 한국해양대학교 컴퓨터공학과(공학사)
2001년 한국해양대학교 컴퓨터공학과(공학석사)
2001년~현재 한국해양대학교 컴퓨터공학과 박사과정
1995년~1996년 미래정보CIM
※관심분야: 데이터베이스, 데이터마이닝, XML

곽용원(Yong-Won Kwak)

1999년 한국해양대학교 컴퓨터공학과(공학사)
2002년 한국해양대학교 컴퓨터공학과(공학석사)
1999년~현재 (주)은성사
※관심분야 : 데이터베이스, XML



박휴찬(Hyu-Chan Park)

1985년 서울대학교 전자공학과(공학사)
1987년 한국과학기술원 전기및전자공학과(공학석사)
1995년 한국과학기술원 전기및전자공학과(공학박사)
1987년~1990년 금성반도체
1997년~현재 한국해양대학교 부교수
※관심분야: 데이터베이스, 모델링방법론, XML