O_2 기반의 XML 문서관리 시스템 설계 및 구현

유 재 수*

Design and Implementation of an XML Document Management System Based on O_2

Jae-Soo Yoo*

Abstract

In this paper, we design and implement a XML management system based on OODBMS that supports structured information retrieval of XML documents. We also propose an object oriented modeling to store and fetch XML documents, to manage image data, and to support versioning for the XML document management system(XMS). The XMS consists of a repository manager that maintains the interfaces for external application programs, a XML instance storage manager that stores XML documents in the database, a XML instance manager that fetches XML documents stored in the database, a XML index manager that creates index for the structure information and the contents of documents, and a query processor that processes various queries.

* 충북대학교 정보통신공학과 교수

1. 서 론

최근 인터넷이 발전함에 따라 웹을 이용한 전자문서의 중요성이 부각되면서 W3C에서 제안된 XML에 대한 다양한 연구가 활발히 진행되고 있다. XML은 이기종간의 시스템에서 작성된 문서의 상호 교환과 다양한 형식의 문서들을 일관성 있게 구조화하기 위하여 고안된 SGML을 웹 상에서 원활히 사용할 수 있도록 간략화시킨 표준안이다[15].

이에 따라 현재 인터넷 Web 문서, 전자도서관, CSCW(Computer Supported Cooperative Work), 그리고 CALS(Commerce At the Light Speed)를 포함한 다양한 분야, 수학 분야의 MathML, 채널 기술의 CDF(Channel Definition Format), 이동통신에서의 WML(Wireless Markup Language) 등에서 XML(eXtensible Markup Language)연구는 상당히 활발하게 진행되고 있으며 대용량의 XML 문서를 효과적으로 저장·관리할 수 있는 시스템의 필요성이 대두되었다[1, 10, 11].

본 논문에서는 XML 문서관리 시스템을 구현하기 위해 다음과 같은 연구를 중점 수행한다. 먼저 XML 문서의 데이터 모델링으로, 이는 XML이 포함하고 있는 많은 특성들을 손실 없이 데이터베이스에 저장하기 위한 것이다. 지금까지 고려되지 않았던 XML 문서 내에 정의된 멀티미디어(이미지 등)까지 관리할 수 있는 데이터 모델링을 제안한다. 특히 제안한 데이터 모델링은 XML 문서의 히스토리 정보를 중요시하는 응용분야를 위해 버전(Version) 정보를 수용할 수 있도록 하였다. XML 문서에 대한 버전닝은 특허 문서 관리, 소프트웨어 설계, 시스템 매뉴얼 등의 응용과 같이 수정된 기존의 문서들이 관리되어야 하는 분야에서 적절히 활용될 수 있다. 두 번째로는 XML 문서의 특성에 기반한 다양한 검색을 지원할 수 있도록 하였다. 기존 연구에서는 대부분이 내용검색, 간단한 구조검색 만을 지원하는데 본 연구에서는 내용 검색과 다양한 구조 검색뿐만 아니라 애트리뷰트 검색과 혼합검색 등을 지원하도록 인덱스 구조를 설계한다. 구현한 XML 문서관리 시스템은 XML 문서의 논리적인 구조정보를 보다 잘 표현할 수 있으며, 객체들에 대한 버전 관리가 용이한 객체지향 데이터베이스의 하나인 O_2를 기반으로 한다.

본 논문의 구성은 다음과 같다. 2장에서는 구조화된 문서의 데이터 모델링에 대하여 기존의 관련 연구 중심으로 알아본다. 3장에서는 설계한 데이터 모델링에 관하여 설명하고, 4장에서는 실제 구현한 XML 문서 관리 시스템에 대하여 알아본다. 그리고 5장에서 성능평가를 한 다음 6장에서 마지막으로 결론 및 향후 연구 방향을 제시한다.

2. 기존 연구

일반적인 문서관리 시스템에 비해 XML 문서관리 시스템은 XML이 갖고 있는 다음과 같은 특성을 고려하여야 한다[4, 5, 11]. XML 문서는 복잡한 구조와 다양한 미디어를 포함할 수 있다. 이러한 XML 문서는 구조 정보를 이용하여 문서를 효율적으로 관리하기 때문에 데이터베이스에 XML 문서, DTD, 구조정보 및 다양한 미디어를 저장, 관리해야 된다. 또한 기존의 정보검색시스템(IRS : Information Retrieval System)은 문서의 논리적 구조에 따른 정보검색을 거의 이용하지 못하고 있다. 그러나 XML 문서의 특성을 볼 때 XML 문서에 대한 내용질의, XML 문서가 갖는 논리적인 계층 구조를 이용한 질의, 엘리먼트가 갖는 속성에 대한 질의 등을 수용할 수 있어야 한다.

현재 요구되는 XML 문서관리 시스템의 기능

을 구분하면 크게 XML 문서의 생성과 배포로 나눌 수 있다. XML 문서 생성에 관련된 주요 기능으로는 XML 문서의 다양한 특성들을 손실 없이 데이터베이스에 반영하기 위한 데이터 모델링 기능과 여러 명의 사용자들이 동일한 문서의 각기 다른 부분에 접근하여 동시에 문서를 작성할 수 있는 공동작업 기능, 상황에 따른 문서 내용의 변화를 참조할 수 있도록 지원하는 버전관리 등을 들 수 있다. XML 문서 배포에 관련된 기능으로는 저장된 문서와 구조정보에 대한 색인 기능과 이를 이용하여 내용 검색이나 다양한 구조 검색을 지원할 수 있는 검색기능, 그리고 응용시스템 개발을 위한 API 등을 들 수 있다.

지금까지 연구되어진 XML 문서관리 시스템을 보면 대부분 XML 문서 배포 쪽에 초점을 맞추어 문서 내용과 구조 정보를 효과적으로 검색하기 위한 데이터 모델링과 이를 이용한 다양한 구조검색의 지원 측면에서 구현되었다[2, 3, 12-14]. 특히 K. Bohm et al.[12]는 SGML 문서의 효과적인 저장을 위해 ODBMS를 설계하였는데, 특히 동적인 DTD를 메타클래스로 구현하는 방법을 보였다. 그러나 XML 문서 내에 정의된 멀티미디어 데이터에 대한 고려가 없으며, XML 문서의 생성에 관한 고려가 전혀 없다. Arnold-Moore et al.[14] GML 문서에 대한 다양한 질의가 가능한 질의 언어를 개발하였지만 실제 구현은 이루어지지 않았다. 또한 SGML 문서를 위한 다양한 데이터베이스 모델들이 제시되었다[13]. 여기서 제시된 모델들은 스키마에서 DTD를 직접 표현하거나, DTD를 위한 스키마를 외부적으로 정의한 것들이다. 이 또한 멀티미디어 데이터에 대한 저장, 관리 및 XML 문서의 생성에 관한 고려가 전혀 없다.

이에 본 논문에서는 XML 문서의 내용뿐만 아니라 문서의 구조정보와 문서 안에 정의된 멀티미디어(특히 이미지) 데이터까지 데이터베이스에 저장하고, XML 문서의 특성을 고려한 다양한 질의를 지원하며, 더 나아가 문서의 수정에 따른 버전닝을 지원할 수 있는 시스템 설계 및 구현을 하고자 한다.

3. O_2 기반의 스키마 설계

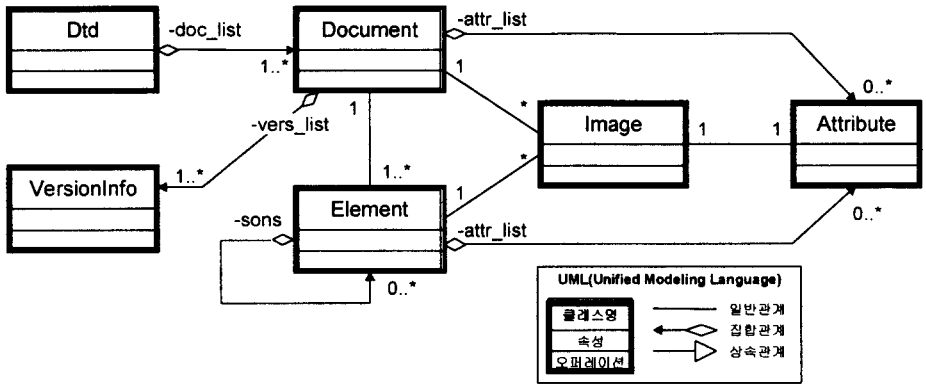
3.1 스키마 설계 목표

본 논문에서는 XML 문서 저장을 위한 스키마를 설계함에 있어서 객체지향 데이터베이스인 O_2 기반의 객체지향 모델링을 사용하였으며, 모든 DTD 문서를 수용할 수 있는 형태의 고정된 스키마를 사용할 수 있도록 정적인 스키마 생성 방법을 고려하였다. 문서 인스턴스 저장 방법에 있어서는 문서를 구조정보별로 쪼개지 않고 저장하는 비분할 저장 모델을 수용하였다. 또한 문서 내에 정의된 이미지 파일에 대하여 데이터베이스에 저장하고 관리하도록 하였으며, 특히 문서단위의 버전을 관리할 수 있도록 하여 새롭게 갱신된 문서가 들어왔을 때 기존의 문서를 지우지 않고 문서 히스토리 정보와 함께 동일한 문서의 새로운 버전으로 등록하도록 하였다. 따라서, 수정되기 이전의 문서도 버전 정보와 함께 언제든지 참조할 수가 있다.

3.2 스키마 설계

객체지향 모델링에서 스키마의 구조는 해당 객체들을 생성하기 위한 클래스들의 집합으로 이루어진다. 그리고, 생성된 객체들간의 관계는 포인터를 이용한 링크로 이루어진다. 스키마를 이루는 각 클래스들의 관계를 UML(Unified Modeling Language) 표기에 의해 나타내면 (그림 1)과 같다.

각 Dtd는 해당하는 Dtd를 갖는 문서들을 가



(그림 1) 구조정보를 위한 각 클래스들 간의 관계

지고, 문서 내에서 구조정보로 추출된 엘리먼트들은 Document 아래에 놓이게 된다. 엘리먼트는 한 개 이상의 하위 엘리먼트를 가질 수 있으며, 엘리먼트내의 애트리뷰트나 애트리뷰트 안에 정의된 이미지를 저장하기 위한 Image 클래스와 연관된다. 그리고, Document들은 버전 정보를 위한 VersionInfo를 갖는다. 스키마 내의 각 클래스들의 구성 요소에 대하여 살펴보면 다음과 같다. 클래스의 구조들은 O_2 데이터베이스 안에 저장되어 있는 형태이며, O_2에서 제공하는 O_2C 형태로 나타내었다.

3.2.1 Dtd 클래스 구조

(그림 2)에서 보여주는 바와 같이 Dtd 클래스는 DTD 파일에 대한 정보를 포함하고 있다. Dtd들간에는 name에 저장된 DTD 파일 이름을 가지고 구별되며, 해당 DTD에 의해 생성된 문서 인스턴스 객체들은 doc_list에 연결된다. dtd_data에는 DTD 파일 내용을 저장하여 언제든지 DTD 파일 내용을 열람할 수 있도록 하였다.

```
class Dtd inherit Object public type
tuple(name: string,
      doc_list: o2_list_Document,
      dtd_data: string)
```

(그림 2) Dtd 클래스 구조

3.2.2 Document 클래스 구조

(그림 3)의 Document 클래스는 문서 인스턴스에 대한 정보를 포함한다. 문서간의 구별은 시스템에서 일괄적으로 제공하는 DID와 XML 문서 파일이름을 갖는 name으로 이루어진다. 또, 해당 DTD를 참조할 수 있도록 dtd_obj에 Dtd 클래스의 객체를 연결하도록 하였고, 최상위에 존재하는 엘리먼트 객체를 참조하기 위해서 root_element를 두었다. image_list는 문서내에 정의되어있는 모든 이미지 객체를 가지고 있어 doc_data안에 있는 문서의 전체 내용을 추출할 때 함께 보낼 수 있도록 하였다.

```
class Document inherit Object public type
tuple(DID: integer,
      name: string,
      vers_list: o2_list_VersionInfo,
      dtd_obj: Dtd,
      root_element: Element,
      attr_list: o2_list_Attribute,
      image_list: o2_list_Image,
      doc_data: string)
```

(그림 3) Document 클래스 구조

3.2.3 Element 클래스 구조

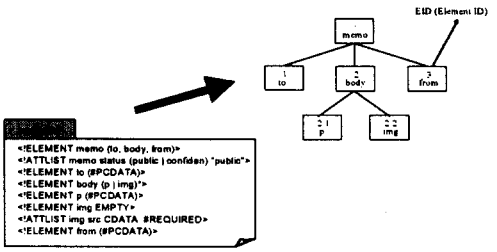
엘리먼트는 문서의 구조를 정의할 때 구조 표

현의 단위로 사용되는 아주 중요한 구성 요소이다. (그림 4)의 Element 클래스에는 상하위 엘리먼트들 간의 참조를 위한 parent와 sons가 있고, 엘리먼트의 문서 내 위치를 나타낼 offset과 length가 있다. 이 offset과 length를 이용하여 Document 클래스 객체 내에 저장된 문서의 일부분을 추출할 수 있다.

```
class Element inherit Object public type
tuple(DID: integer,
      EID: string,
      document_obj: Document,
      name: string,
      type: integer,
      offset: integer,
      length: integer,
      attr_list: o2_list_Attribute,
      image_list: o2_list_Image,
      parent: Element,
      sons: o2_list_Element)
```

(그림 4) Element 클래스 구조

XML 문서를 저장함에 있어서 생성된 객체들 간의 식별을 용이하게 하기 위해서 Document 객체를 위한 DID와 Element 객체를 위한 EID를 사용하였다. DID는 Document 객체를 생성할 때 시스템에서 일괄적으로 부여하도록 하여 문서마다 유일한 값을 갖도록 하였다. EID는 문서 내에 특정 엘리먼트를 구별하면서 엘리먼트 간의 계층정보를 표현할 수 있다. EID는 DTD의 논리적 구조를 분석한 후 각 엘리먼트 타입에 부여되는 유일한 값이다. 이 EID를 부여하는 방법은 UNIX 파일시스템에서 디렉토리를 표현하는 방법을 사용한다. 즉, root 엘리먼트는 '/'로 표현되며 root 엘리먼트에 포함되는 엘리먼트는 '/1'나 '/2'와 같이 표현된다. 다음 (그림 5)는 문서구조를 트리로 표현하고 각 엘리먼트에 EID를 할당한 예이다.



(그림 5) EID의 할당

3.2.4 Attribute 클래스 구조

(그림 6)의 Attribute 클래스는 엘리먼트 내에 정의된 애트리뷰트의 name과 value를 가지고 있다. 애트리뷰트의 값은 타입에 관계없이 문자열로 저장하도록 하였으며, type에 의해 구별하도록 하였다. 또, 애트리뷰트의 값으로 정의된 이미지의 객체를 참조할 수 있도록 하였다.

```
class Attribute inherit Object public type
tuple(name: string,
      type: integer,
      value: string,
      element_obj: Element,
      image_obj: Image)
```

(그림 6) Attribute 클래스 구조

3.2.5 Image 클래스 구조

(그림 7)의 Image 클래스는 이미지 파일 이름과 내용을 포함하고, 이미지 파일이 정의된 엘리먼트와 애트리뷰트 객체를 참조할 수 있도록 하였다.

```
class Image inherit Object public type
tuple(name: string,
      type: integer,
      element_obj: Element,
      attribute_obj: Attribute,
      image_data: string)
```

(그림 7) Image 클래스 구조

3.2.6 VersionInfo 클래스 구조

(그림 8)의 VersionInfo는 문서들의 버전 정보를 갖는 객체를 생성하기 위한 클래스이다. VersionInfo 안에는 이 버전의 문서가 지워졌는지를 나타내는 removed와 갱신한 사용자의 정보, 수정 날짜, 변경 내용 등을 포함한다.

```
class VersionInfo inherit Object public type
tuple(removed: integer,
      version_ID: integer,
      user: string,
      date: integer,
      content: string)
```

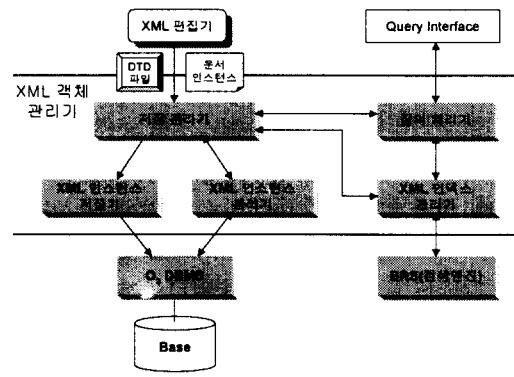
(그림 8) VersionInfo 클래스 구조

4. O_2 기반의 XML 문서관리 시스템 구현

4.1 O_2 기반의 XML 문서관리 시스템 구조

본 논문에서는 (그림 9)에서와 같이 XML 편집기와 데이터베이스 사이에 있으면서 편집기로부터 들어오는 DTD 파일과 문서 인스턴스를 데이터베이스에 저장하고, 요청된 문서의 내용을 보내주는 기능을 수행하는 XML 객체 관리기를 구현하였다. 시스템의 구현환경으로는 Ardent사의 O_2 데이터베이스 버전 5.0과 BRS/Search 버전 6.3 시스템을 이용하였으며 SUN Solaris 2.5.1 환경에서 C++ 언어를 이용하여 구현하였다. 구현한 시스템의 각 모듈 별 기능들은 다음과 같다.

저장 관리기는 XML 편집기에서 들어오는 DTD 파일과 문서 인스턴스를 받아서 XML 인스턴스 저장기를 통해 저장하고, 요청된 문서에 대하여 XML 인스턴스 관리기를 통해 데이터를 추출하여 보내주는 역할을 담당한다. XML 인스턴스 저장기는 저장 관리기로부터 DTD 문서와 XML 문서 인스턴스를 받아 데이터베이스에 직접 저장하는 역할을 담당한다. 먼저, DTD를 위한 Dtd 객체와 문서 인스턴스를 위한 Document 객체를 생성하고, 문서 인스턴스 내에 정의되어 있는 구조 정보들을 각각 추출하여 객체를 생성하고 데이터베이스에 저장한다. XML 인스턴스 관리기는 문서 인스턴스 내용의 추출과 삭제를 담당한다. 저장 관리기에서 요청된 문서를 데이터베이스에서 찾아 문서 전체 또는 일부분의 내용을 넘겨주는 기능과 해당 문서와 그와 관련된 모든 요소들을 데이터베이스에서 삭제하는 기능을 담당한다. XML 인덱스 관리기는 BRS 검색엔진을 이용하여 문서 인스턴스와 구조정보로부터 인덱스 term을 추출하여 인덱스를 생성하고 질의처리기에 색인정보를 제공한다. 질의처리기는 색인기에 의해 생성된 인덱스를 이용하여 XML문서의 내용 및 구조검색을 수행한다.



(그림 9) XML 문서관리 시스템의 구성도

4.2 문서 인스턴스의 저장

최초에 저장 관리기를 통해 DTD와 문서 인스턴스가 들어오게 되면 저장관리기는 먼저 XML 인스턴스 관리기를 통해 데이터베이스에 동일한 문서가 저장되어 있는지 확인한다. 동일한 문서의 존재를 확인하는 이유는 기존의 문서를 지우

고 새로 들어온 문서를 저장할 것인지, 아니면 동일한 문서의 새로운 버전으로 저장할 것인지를 저장 관리기가 결정하기 위해서이다. 기존의 문서를 지우고자 한다면 XML 인스턴스 관리기를 통해 해당 문서를 삭제하고 XML 인스턴스 저장기를 호출하게 된다. 기존의 문서를 그대로 보관하고자 한다면 XML 인스턴스 저장기에 버전 정보와 함께 DTD와 문서 인스턴스를 넘겨 저장하도록 한다.

```
<!DOCTYPE memo SYSTEM  memo.dtd >
<memo status="public">
<to>홍길동</to>
<body>
<p>안녕하오.</p>
<img src= map.jpg >
<p>자비령에서 봅세.</p>
</body>
<from>장길산</from>
</memo>
```

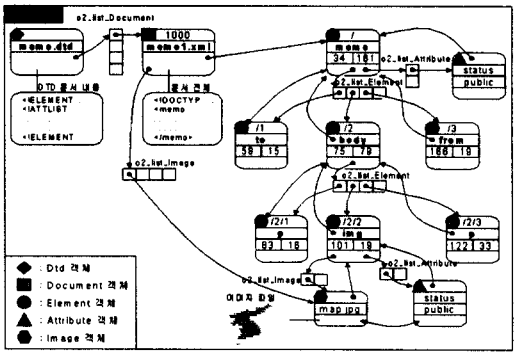
(그림 10) sample.xml

XML 문서를 위한 객체를 생성할 때 시스템으로부터 DID를 받아 부여한다. 그리고, 해당 문서를 파싱하면서 내부에 정의된 엘리먼트들을 추출하여 EID를 부여하고, 서로간의 관계를 링크로 연결한다. 또한, 엘리먼트 내부에 애트리뷰트가 정의되어 있으면 이를 생성하여 저장하고, 이미지를 포함하고 있으면 해당 객체를 생성하고 이미지 파일을 데이터베이스로 읽어온다. 실제 앞장에서 예로 든 memo.dtd를 가지고 (그림 10)과 같이 sample.xml 문서를 작성하여, 데이터베이스 내에 저장한 구조는 (그림 11)과 같다.

4.3 문서 인스턴스의 추출과 삭제

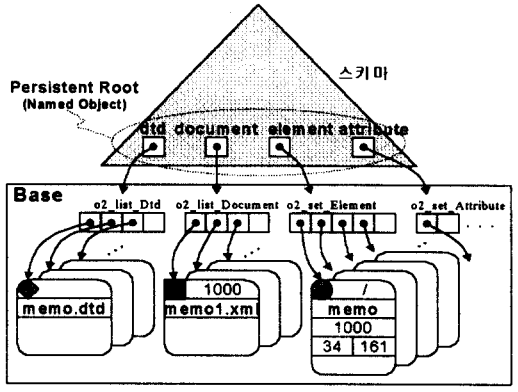
4.3.1 문서 전체 내용의 추출

데이터베이스에서 문서 전체의 내용을 추출



(그림 11) 데이터베이스에 저장된 문서 인스턴스의 객체들

하고자 할 때 먼저 원하는 문서의 이름이나 문서의 DID를 알고 있어야 한다. 문서 이름이나 DID가 XML 인스턴스 관리기에 들어오면 이것을 가지고 XML 인스턴스 관리기는 OQL을 이용하여 데이터베이스에서 해당 객체를 찾는다. 그런 다음, 찾은 문서 인스턴스 객체내의 메소드를 이용하여 문서의 내용을 추출하게 된다.



(그림 12) persistent root를 이용한 객체 접근

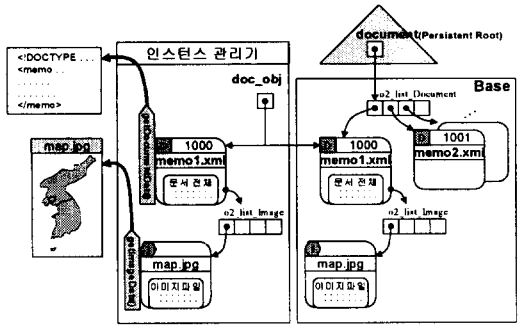
(그림 12)는 O_2에서 사용하는 Persistent Root를 이용한 데이터베이스 내 객체에 대한 접근을 나타내고 있다. O_2에서는 데이터베이스 안에 저장된 객체들을 최초에 접근하기 위해서 Persistent Root 또는 named object라고 하는 엔트리 포인터를 사용한다. 데이터베이스 내에 생성된

모든 객체들은 Persistent Root와 직·간접적으로 연결되어 있어야 해당 객체의 접근이 가능하다. 이 엔트리 포인터는 스키마를 생성할 때 스키마 안에 등록하여 사용하며, 새로 생성된 객체들은 persistent root의 콜렉션에 연결시켜 OQL에서 해당 객체를 찾을 때 이용한다[6, 8].

```
d_Ref<Document> doc_obj;
d_OQL_Query get_doc("element(select d from d in document where d.getName == $1)");
get_doc << "memo1.xml";
d_oql_execute(get_doc, doc_obj);
```

(그림 13) Document 객체를 얻기 위한 OQL의 사용

(그림 13)은 문서 파일의 전체 내용을 열람하기 위하여 해당 Document 객체를 찾는 OQL 사용의 예를 보이고 있다. get_doc이란 질의 객체를 생성하고 질의를 수행하게 되면, Persistent Root인 document안의 콜렉션에서 'memo1.xml'을 이름으로 갖는 Document 객체를 찾아준다[7]. 얻어진 객체에서 문서의 전체 내용을 추출하는 과정을 살펴보면 (그림 14)와 같다.



(그림 14) Document 객체에서 문서 전체의 내용을 추출하는 과정

위의 그림에서와 같이 OQL 결과로 해당 Document 객체가 doc_obj로 넘겨진다. 여기에서 doc_obj는 d_Ref<Document>라는 persistence pointer로 사용되는데 persistence pointer에 연결된 객체는 어플리케이션의 메모리에 자동적으로 load되고 데이터베이스 안의 객체와 동일하게 사용된다. doc_obj에 넘겨진 객체는 메소드인 getDocumentData()를 이용하여 문서의 내용을 스트링으로 넘겨준다. 그리고, 문서 안에 정의되어 있는 이미지 파일들이 데이터베이스에 존재하므로 Document 객체에 연결되어 있는 Image 객체를 참조하여 이미지 파일들을 가져오도록 한다.

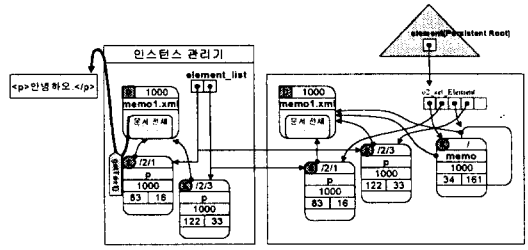
4.3.2 엘리먼트 기반의 문서 내용의 일부분 추출

특정 엘리먼트의 내용을 참조하고자 할 때 먼저 해당 엘리먼트의 객체를 OQL을 이용하여 검색하여야 한다. (그림 15)는 데이터베이스 안에 저장되어 있는 모든 엘리먼트들 중에서 엘리먼트 이름이 'p'인 엘리먼트들을 찾기 위한 OQL의 사용 예이다. 아래의 (그림 16)은 OQL을 이용하여 검색된 엘리먼트들을 가지고 문서 내용의 일부분을 추출하는 과정을 나타내고 있다.

```
d_List<d_Ref<Element> > element_list;
d_OQL_Query get_el_list("select e from e in element where e.getName == $1");
get_el_list << "p";
d_oql_execute(get_el_list, element_list);
```

(그림 15) 특정 엘리먼트을 얻기 위한 OQL 사용의 예

각각의 엘리먼트들은 자신이 속한 문서 인스턴스의 Document 객체를 포인트하고 있어, 자신이 가지고 있는 오프셋과 길이 정보를 가지고 Document 객체를 직접 찾아가서 문서의 내용을 가져올 수가 있다. 포인터를 이용한 직접 접



(그림 16) 문서 내용의 일부분을 추출하는 과정

근을 사용하기 때문에 문서 내용의 추출을 더욱 빨리 수행할 수 있다. 그리고 엘리먼트 아래에 이미지 객체가 존재하면 함께 보내줄 수 있도록 하였다.

4.3.3 문서 인스턴스의 삭제

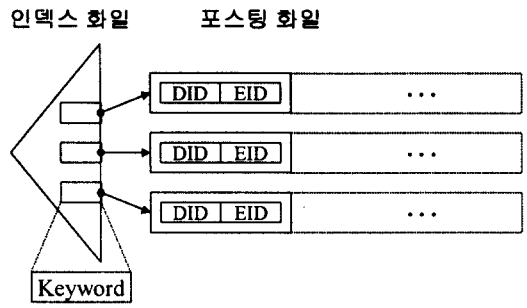
데이터베이스내의 객체들의 삭제는 비분할 모델의 특성상 문서 인스턴스 단위로 이루어진다. 삭제할 문서의 이름이나 DID가 들어오면 OQL을 이용하여 해당 문서의 Document 객체를 찾아 제거하고, 그에 속한 모든 엘리먼트, 애트리뷰트, 이미지 객체들도 제거한다.

4.4 인덱스의 구성

XML 인덱스 관리기는 XML 문서에 대한 내용, 구조, 애트리뷰트 검색과 이들이 혼합된 형태의 혼합검색을 지원하기 위하여 색인을 생성하고 관리한다. XML 인덱스 관리기에서 생성하는 색인에은 본문 검색을 위한 내용 색인, 구조 검색과 혼합검색을 지원하는 구조 색인, 그리고 애트리뷰트 검색을 위한 애트리뷰트 색인 등이 있다.

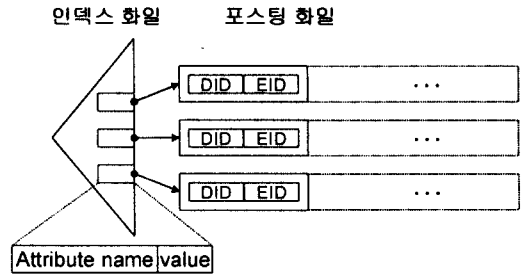
내용 색인은 문서 전체의 내용에 대하여 태그 정보가 제거된 전문(full-text)을 가지고 BRS 검색엔진을 이용하여 인덱스 term을 추출한다. 추출된 키워드로 색인을 구성하고 포스팅 파일에는 문서 인스턴스와 매핑할 수 있는 DID를 둔다.

구조 색인은 구조의 단위가 되는 엘리먼트가 중심이 되며 저장관리기로부터 엘리먼트의 내용을 추출한 다음 태그정보를 삭제하고 인덱스 term을 추출한다. (그림 17)에서와 같이 구조 색인의 포스팅 파일은 개개의 엘리먼트와 매핑할 수 있는 정보인 DID, EID로 구성된다.



(그림 17) 구조색인의 구조

(그림 18)은 애트리뷰트의 색인 구조를 나타낸다. 애트리뷰트 색인은 애트리뷰트 이름과 값으로 인덱스가 구성되고, 해당 애트리뷰트가 포함되어 있는 엘리먼트와 매핑할 수 있도록 DID와 EID로 구성된 포스팅 파일이 존재한다.



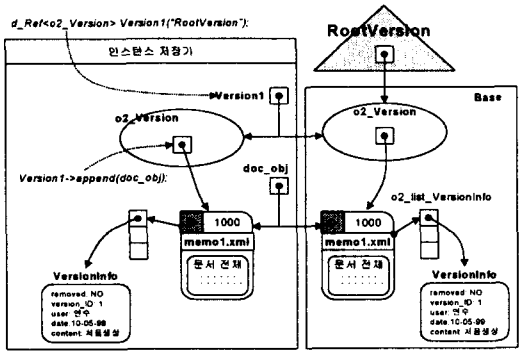
(그림 18) 애트리뷰트 색인 구조

4.5 문서의 버전 지원

본 논문에서는 O_2 데이터베이스에서 지원하는 버전 기능을 이용하여 문서 인스턴스 단위의 버전정보를 지원하도록 하였다. 문서 인스턴스의 버전닝 과정을 보면 (그림 19)와 같다.

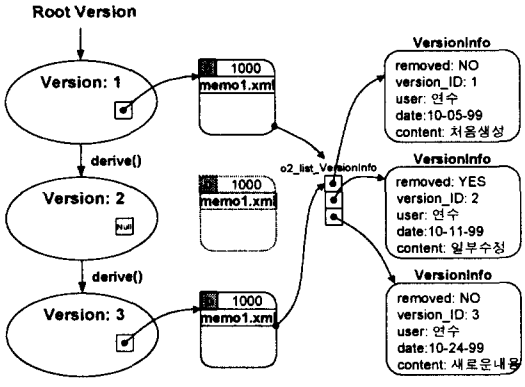
처음 새로 생성된 문서 인스턴스는 버전 1을 부여받고 버전 정보를 사용자로부터 입력받는다. 나중에 동일한 이름의 문서 인스턴스가 들어오게 되면 새로운 버전에 추가할 것인지 아니면, 기존 문서를 삭제하고 새로 삽입할 것인지

를 판단한다.



(그림 19) 문서 인스턴스의 버전닝 과정

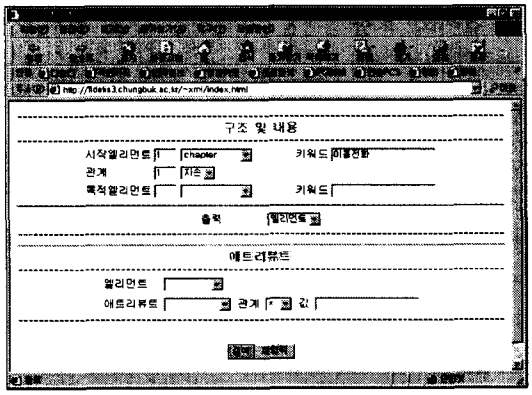
O_2에서 버전을 사용하기 위해서는 먼저 o2_Version 객체를 생성하여야 한다. 그리고, 버전을 부여할 객체를 버전 객체에 append 시키면 된다. 새로운 버전이 필요로 하게 되면 기존의 버전 객체를 derive해서 새로운 버전 객체를 생성한 다음, derive할 때 상속된 문서 인스턴스 객체를 새로운 문서 인스턴스로 대체하면 된다. (그림 20)에서와 같이 버전 객체내의 문서 인스턴스가 삭제되더라도 버전 정보는 삭제되지 않고 계속해서 데이터베이스에 해당 문서 인스턴스에 대한 히스토리 정보로 존재하게 된다[9].



(그림 20) 문서 인스턴스의 버전 정보 관리

4.6 사용자 인터페이스

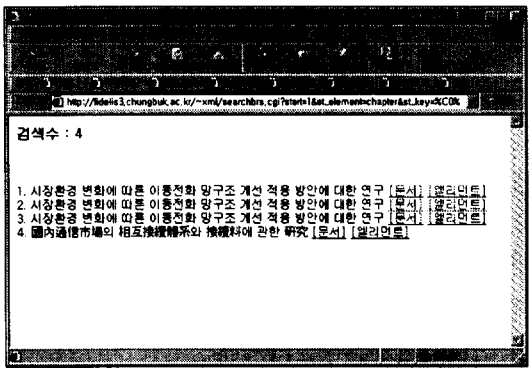
사용자가 XML 문서에 대한 검색을 하기 위해서는 사용자 인터페이스를 필요로 한다. 사용자 인터페이스는 기본적으로 문서의 내용에 대한 질의뿐만 아니라 구조 질의와 이들 둘을 혼합한 혼합질의를 원활히 처리하여 사용자에게 보여줄 수 있어야 한다. 본 논문에서는 XML 문서 저장시스템을 테스트하기 위해 사용자 인터페이스를 웹을 이용한 CGI로 구현하였다. (그림 21)에서와 같이 사용자 인터페이스는 크게 구조 및 내용 검색 부분과 애트리뷰트 검색 부분으로 나누어진다. 내용 검색은 키워드 부분에 해당 단어를 입력하고 검색을 수행한다. 구조 및 혼합 검색은 시작 엘리먼트를 기점으로 목적 엘리먼트와의 관계를 나타내어 수행한다. 엘리먼트 간의 관계에는 조상, 형제, 자식 등이 있다. 애트리뷰트 검색은 애트리뷰트 값의 다양한 관계 연산을 제공한다.



(그림 21) 사용자 인터페이스

(그림 21)은 "이동전화를 포함하는 첫 번째 chapter의 자식 엘리먼트를 찾아라"라는 구조와 내용이 혼합된 혼합 질의의 예를 보여주고 있다. (그림 22)는 검색의 결과로서 문서 전체

또는 해당 엘리먼트들을 참조할 수 있도록 하였다.



(그림 22) 질의 결과

5. 성능 평가

구현한 XML 문서 저장관리 시스템의 성능평가를 위하여 다음과 같은 과정을 수행하였다.

- 적용분야 선정 : 학회 논문지
- DTD 파일 작성 : 학회 논문지 형식에 맞도록 설계
- 문서 인스턴스 생성 : DTD 파일에 맞는 XML 문서 파일 생성(350건)
- DTD와 문서 인스턴스의 저장 : 생성된 문서 인스턴스를 데이터베이스에 저장
- 테스트 : 사용자 인터페이스를 이용하여 문서가 올바르게 저장되었는지 검색

테스트의 결과로서 본 논문에서는 정보 검색 시스템의 일반적인 평가 방법으로 사용되는 재현율(recall)과 정확율(precision)을 이용하였다. 재현율은 문서 집합에서 사용자가 원하는 문서를 어느 정도 검색하였는가를 나타내고, 정확율은 검색된 문서들 중에서 사용자가 원하는 문서가 얼마나 포함되어 있는가를 나타낸다. 재현율과 정확율을 구하는 수식은 아래와 같다.

$$\text{재현율} = \frac{\text{검색된적합문서수}}{\text{적합문서총수}}$$

$$\text{정확율} = \frac{\text{검색된적합문서수}}{\text{검색된문서총수}}$$

평가에 사용된 질의 예는 다음과 같다.

1. "abstract"이라는 엘리먼트에 "정보"라는 단어를 포함하는 문서를 검색하시오.
2. "분산"이라는 단어를 포함하는 "title" 엘리먼트의 상위 엘리먼트를 검색하시오.
3. 논문이라는 단어를 포함하는 두 번째 title 엘리먼트를 검색하시오.
4. "데이터"라는 단어를 포함하는 첫 번째 "section"의 자식 엘리먼트를 검색하시오.
5. 애트리뷰트의 이름이 "lineWidth"이고 값이 "100"인 엘리먼트를 검색하시오.

위 질의에 대한 재현율과 정확율을 각각 계산하면 <표 1>과 같다. <표 1>에서 보듯이 XML 문서 인스턴스와 구조정보들이 정확하게 데이터베이스에 저장됨을 알 수 있다.

〈표 1〉 질의에 대한 재현율과 정확율

질 의	재현율	정확율
1번 질의	1	1
2번 질의	1	1
3번 질의	1	1
4번 질의	1	1
5번 질의	1	1

6. 결 론

본 논문에서는 XML 문서의 구조적인 특성을 효과적으로 데이터베이스에 반영할 수 있는 객체지향 모델링과 이를 이용한 XML 문서 관리 시스템의 설계 및 구현을 수행하였다. XML 문서의 데이터 모델링을 함에 있어서 구조 정보를

보다 쉽게 표현하기 위하여 OODB인 O_2를 기반으로 하는 객체지향 모델링을 사용하였다. 또한, XML 문서의 추출 성능향상을 위해 XML 문서의 전체를 저장하는 비분할 모델, 그리고, 모든 DTD 문서를 수용할 수 있는 형태의 정적인 스키마 생성을 특징으로 한다.

XML 문서의 효율적인 저장, 관리 및 다양한 검색을 위하여 저장관리기, XML 인스턴스 저장기, XML 인스턴스 관리기, XML 인덱스 관리기, 질의처리기 등의 세부 모듈들을 구현하였다. 특히 XML 문서내에 정의된 이미지 파일에 대한 데이터베이스로의 저장 및 추출을 구현하였고, 문서 단위의 버전기능을 추가하여 문서의 히스토리 정보를 중요시하는 응용분야에 유용하게 사용될 수 있도록 하였다. 마지막으로 구현한 XML 문서 관리 시스템의 검색 효율을 평가하기 위해 재현율과 정확율을 이용하여 평가한 결과를 보면 XML 문서에 대하여 구조에 기반한 질의를 모두 만족함을 알 수 있다.

향후 연구 과제로는 문서 단위의 버전 지원뿐만 아니라 엘리먼트 단위의 버전을 지원할 수 있도록 XML 문서관리 시스템을 확장하는 것이다.

참 고 문 헌

[1] 김준하, 김태현, 이해용, 김승배, "RDBMS 기반의 SGML Repository Manager의 개발", 한국정보과학회 추계 학술발표논문집 5권 2호, p.839-842, 1998.

[2] 박철현, 정재현, 심대익, 이상구, "구조화된 문서에 대한 DBMS와 IRS의 성능 비교", '99 한국데이타베이스 학술대회 논문집 15권 1호, p.218-225, 1999.

[3] 연제원, 장동준, 김용훈, 이강찬, 이규철, "효율적인 검색 지원 SGML 저장 관리기의 설계 및 구현", '99 한국 데이터베이스 학술대회 논문집 15권 1호, p.136-143, 1999.

[4] 유재수외 8인, "전자도서관 표준 문서 관리를 위한 XML 저장관리기 개발", (주)한국지식웨어, 시스템매뉴얼, 충북대학교 컴퓨터정보통신연구소, 1999.

[5] 유재수외 4인, "XML 저장관리기 기술개발에 관한 연구", 넥스텍 객체기술 R&D 연구소, 최종연구보고서, 충북대학교 산업과학 기술연구소, 1999.

[6] Ardent, *"ODMG C++ Binding Guide,"* O_2 Database Reference Manual, p.95-115, 1998.

[7] Ardent, *"ODMG OQL User Manual,"* O_2 Database Reference Manual, p.57-90, 1998.

[8] Ardent, *"O_2 System Administration Reference Manual,"* O_2 Database Reference Manual, 1998.

[9] Ardent, *"O_2 Version Reference Guide,"* O_2 Database Reference Manual, p.61-100, 1998.

[10] Brian Lowe, Justin Zobel, Ron Sacks-Davis, "A Formal Model for Databases of Structured Text," DASFAA, p.449-456, 1995.

[11] Francois, "Generalized SGML repositories : Requirements and modelling," Computer Standards & Interfaces, pp.11-24, 1996.

[12] K. Bohm and K. Aberer, "Storing HyTime Document in An Object-Oriented Database," In Proc. of CIKM '94, pp.26-33, 1994.

[13] R. Sacks-Davis, T. Arnold-Moore, and J. Zobel, "Database systems for structured documents," In Proc. of the International

Symposium on Advanced Database Technologies and Their Integration (ADTI '94), Nara, Japan, pp.277-283, 1994

[14] T. Arnold-Moore, M. Fuller, B. Lowe, J. Thom and R. Wilkinson, "The ELF Data Model and SGQL Query Language for Structured Document Database," In Proc. of the Australian Database Conference, pp. 17-26, 1995.

[15] W3C, *"Extensible Markup Language(XML) 1.0,"* http : //www.w3.org/TR/1998/REC-xml-19980210.html, 1998.

■ 저자소개

유 재 수

전북대학교 컴퓨터공학과를 졸업하고, KAIST에서 공학석사, 그리고 KAIST에서 공학박사학위를 취득하였으며, 목포대학교에서 전임강사로 재직하였다. 현재 충북대학교 공과대학 전기전자공학부 조교수로 재직하고 있으며 주요관심분야는 데이타베이스 시스템, 정보검색, 멀티미디어 데이타베이스, 분산객체 컴퓨팅 분야이다.