XML을 적용한 표준 문서 관리 시스템의 설계 및 구현*

Design and Implementation of Standard Document Management System

이 준 섭(Jun-Seob Lee)*** 유 정 연(Jeong-Youn Yu)**
권 석 훈(Seok-Hun Kwon)** 나 재 열(Jae-Youl Na)**
이 규 철(Kyu-Chul Lee)** 구 경 철(Kyoung-Chul Koo)***
박 기 식(Ki-Shik Park)*** 박 치 항(Chi-Hang Park)***

목 차

초 록

급속한 과학 기술의 발달로 인해 상호간의 정보교환의 요구는 증가하게 되었으나 서로 다른 시스템 환경으로 인해 정보 교환에 많은 문제점이 발생하였다. XML 기반의 정보 교환은 이를 위한 해결하기 위한 방안이며 여러 연구자들이 공동으로 의견을 교환하여 작성해야 하는 표준 문서의 관리에 XML을 적용하면 매우 효과적이다 본 논문에서는 표준 문서 제정 과정에서 이루어지는 문서의 공유 및 상호 교환을 보다 생산적이며 효율적으로 제공하기 위해 기존 시스템 환경의 변화 없이 차세대 인터넷 문서의 표준인 XML을 기반으로 문서를 교환하며 이를 효과적으로 저장, 검색, 관리 할 수 있는 시스템 모델을 설계 및 구현하였다.

ABSTRACTS

The Request of the information exchange is increasing because of the advanced rapid science and technology. But a different system environment has occurred many problems on the information exchange. The information exchange on based XML is a solution to the problem. It takes effect in the standard document management application that is make standard document to cooperate with many researchers mutually. This paper is design and implementation of system model for efficient exchange, store, search and manage document on based XML document in established course of standard document.

키워드: XML, 표준 문서, 메타데이타, hwp, DTD, 정보통신단체표준문서

* 본 논문은 2000년도 한국전자통신연구원의 표준 관련 문서 교환을 위한 XML-DTD 및 관련 Repository 기술 연구의 위탁과제로 수행된 결과임

** 충남대학교 공과대학 컴퓨터공학과(jyyou@ce.cnu.ac.kr)
Dept. of Comupter Engineering, Chungnam National University

*** 한국전자통신연구원 정보화기술연구본부
Protocol Engineering Center Electronics and Telecommunications Research Institute

접수일자 2000년 11월 20일

1. 서 론

하드웨어, 운영체제, 네트워크 등 시스템의 발달과 함께 이를 기반으로 한 다양한 형태의 컴퓨터 응용 개발은 개인과 개인, 기업과 기업, 그리고 사회 전반에 걸친 업무 관리 및 개발 환경에 많은 영향을 미치고 있다. 이러한 변화는 과학 기술은 물론, 업무 처리에 관련된 정보의 급속한 증가를 초래하였으며 이로 인해 발생하는 각 관련 정보들간의 효과적인 정보 교환 기능을 요구하게 되었다.

그러나 기존의 시스템 환경에서는 운영 환경, 응용 프로그램 등 업무 환경이 서로 다름으로 인해 정보의 교환 시 응용 프로그램의 변경, 데이터의 변환 등 다양한 문제점을 안고 있다. 워드 프로세서의 경우 서로 다른 업체에서 제공하는 툴 사용, 같은 툴에 대한 상이한 버전 지원 등 여러 가지의 원인으로 인해 정보 교환의 어려움이 발생한다. 이러한 문제점들은 현재뿐만 아니라, 향후 업무 개발 시간 및 환경에 많은 장해 요인으로 작용하고 있다.

XML 기반의 정보 관리는 문제점들을 해결하기 위한 유일한 방안으로 소프트웨어 및 플랫폼에 중립적인 전자 문서 관리를 제공함으로써 정보 교환이 빈번한 시스템 관리에 적용되고 있으며 현재 이를 이용한 시스템 개발이 활발히 이루어지고 있다. 특히, 상호간의 정보 교환 문서들 중 표준 문서는 국가기관, 산업체, 연구소, 학회 등 해당 분야의 단체 표준을 제정하여 정보를 공동활용하며 각 분야의 관련 제품 및 서비스 등의 호환성과 연동성을 제공하는 중요한 역할을 담당하는 것으로써 여러 관련 연구자들이 의견을 교환하여 작성해야 하는 표준 개발에 XML 기반 문서 관리 기술을 적용하여 표준화활동의 효율성을 높일 수 있다[6].

본 논문에서는 인터넷 중심의 정보 교환 사회에서 보다 생산적이며 효율적인 표준 관련 문서의 공유 및 상호 교환을 위해 차세대 인터넷 문서의 표준인 XML을 기반으로 문서를 교환하며 이를 효과적으로 저장, 검색, 관리할 수 있는 시스템 모델을 제안하였으며 이를 기반으로 시스템을 설계 및 구현하였다.

본 논문에서 구현한 표준 문서 관리 시스템은 다음과 같은 특징을 가지고 있다. 첫째, 기존의 시스템 환경에 영향을 미치지 않고 보다 효과적인 정보 교환을 위해 자동화된 XML 기반의 문서 정보 관리를 제공하고 있다. 또한, 이제까지 XML 형태의 정보 관리를 위해서는 요구되어 왔던 XML 전용 에디터를 기반으로 한 문서 작성 방법을 탈피하여 XML의 기반지식이 없는 일반사용자들도 XML 기반의 문서 관리를 쉽게 할 수 있도록 하였다. 현재, 이러한 형태의 정보 관리는 국내·외적으로 큰 관심이 대상이 되고 있으며 이를 위한 연구가 진행 중에 있다.

둘째, 효과적인 문서 정보 교환을 위해 사용자의 권한에 따른 자동화된 파일 문서의 변환 기능을 제공하고 있다. 특히, 특정 단체 표준 문서 제정 과정에서는 사용자의 권한에 따른 문서 파일 제공이 중요시되고 있다. 본 시스템에서는 사용자의 권한에 따라 수정, 편집이 가능한 문서 형태와 수정이 불가능한 PDF 형태의 문서를 제공하도록 하였다.

본 논문의 전체 구성은 다음과 같다. 제 2장은 표준 문서 관련하여 현재 진행 중인 기술

들을 조사하였다. 제 3장에서는 XML 기반의 문서 관리를 위한 DTD 설계를 하였으며, 제 4장과 제 5장을 통해 전체 시스템의 설계 및 구현을 기술하였다. 제 6장에서는 결론을 기술하였다.

2. 관련 기술 동향

인터넷의 발달과 함께 상호간의 웹을 기반으로 한 상호간의 문서 정보 교환을 위한 서비스가 활발히 진행 중에 있으며 이를 효과적으로 전달하고 작업하기 위한 시스템 개발되고 있다. 이 중 각 분야의 표준 문서는 향후 시스템간의 정보 교환 및 시스템 개발을 원활하게 하기 위한 기초 정보를 제공하는 것으로 이러한 기능을 제공하는 표준 문서를 효과적으로 빠르게 작업하고 공유하기 위한 연구가 진행 중에 있다. 이에 본 논문에서는 현재 국내·외적으로 진행 중인 연구 중 대표적인 연구로서 Invisible World, Inc에서 수행하고 있는 인터넷 문서 관리를 위한 RFC 관리 시스템과 한국과학재단에서 2000년부터 진행중인 과제로서 일반적인 과제 제안서에서 보고서까지 전반적인 행정 문서를 전자화하여 처리하기 위한 연구를 소개한다.

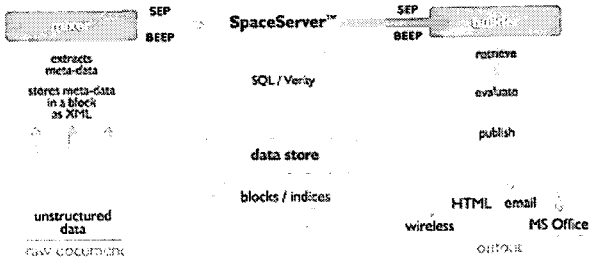
2. 1 RFC(Request For Comments)[2][3]

RFC는 설명 요청, 자료 요청이란 뜻이며 1969년 이후 인터넷에 관련된 여러 가지 내용들을 기록해 놓은 일련의 문서들을 지칭하는 용어로 인터넷과 TCP/IP 사용서를 말한다. 이는 문서 번호와 함께 표기를 하며 인터넷 통신 규약과 여러 가지 실험 결과들을 자료로 만들어 놓은 것으로 인터넷 관련 문서를 체계적으로 관리함으로써 인터넷 관련 정보들의 오류들을 최소화하기 위한 목적으로 가지고 있다.

Invisible Worlds, Inc의 Marshall T.Rose는 XML을 기반으로 하여 I-D(Internet-Draft)와 RFC series를 위한 문서 작성 정의를 제안하였으며, <그림 1>과 같이 XML 기반의 메타데이타 관리를 위한 통합 솔루션인 Blocks 프로토콜 및 구조를 만들어왔다.

2. 2 한국 과학 재단

최근에 한국 과학 재단에서는 전자 문서의



<그림 1> BlocksTM 구조

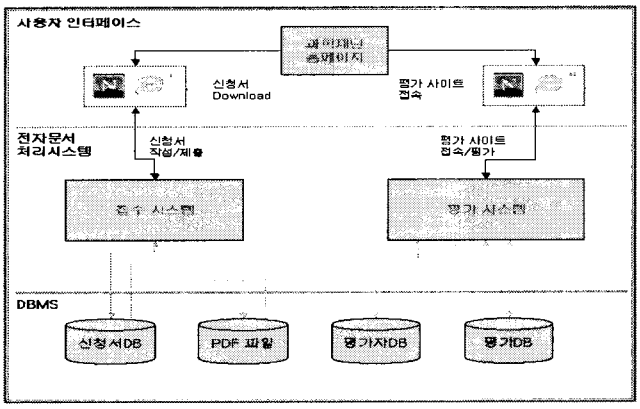
유통 방안 연구 과제로서 〈그림 2〉와 같이 제안서의 제출에서 평가까지의 처리 과정에서 이루어지는 업무 처리를 자동화하기 위해 기초가 되는 파일럿 프로젝트를 수행하였다. 이 파일럿 프로젝트는 '우수 여성 과학자 도약 지원 연구' 과제에 대한 신청서 접수 및 평가 작업을 대상으로 수행하고 있으며, 지금까지 매뉴얼이나 우편으로 이루어져 왔던 처리 작업을 온라인으로 관리 및 처리함으로써 보다 효과적이며 빠른 업무 처리를 수행하고 있다. 이 프로젝트는 웹을 통하여 연구 제안자들로부터 접수받은 워드 프로세서 형태의 제안서를 XML형태의 문서로 변환 및 관리하고 제안서에 대한 평가를 위해 저장된 문서에 대한 검색 및 평가 정보 처리 작업을 수행하였다.

3. DTD 설계

Hwp나 MS Word등 일반 워드 프로세서를 이용하여 작성한 문서들은 바이너리 파일들로 저장된다. 이러한 바이너리 파일들의 내용들을 서로 교환하여 분석, 처리하는 것은 그리 쉬운 일이 아니다. 워드 프로세서에서는 이를 위한 해결 방안으로 서로 다른 운영체계 내에서 운영되는 워드 프로세서들 간의 텍스트 파일을 교환하기 위한 파일 형식을 제공한다. 현재, Hwp에서는 HWPML 파일 저장 형식(화일 이름.hwpml)과 MS Word에서는 RTF 파일 형식(파일 이름.rtf)을 제공하고 있다. 이러한 파일 형식을 이용한 문서의 분석 및 처리는 XML 문서 생성에 있어서 효과적인 방안을 제공한다. 본 장에서는 이를 위해 필요한 문서의 서식 설정 및 DTD 설계에 대해 기술한다.

3. 1 서식 설정

Hwp의 HWPML파일이나 MS Word의 RTF파일은 다음 〈그림 3〉과 같이 글자 크기, 폰트, 스타일 정보 등 문서의 작성 시 이용되는 서식의 정보 등을 제공한다. 이러한 정보를 보다 효율적으로 이용하여 문서에 대한 의미를 정확하게 분석하고 처리함으로서 XML문서를 생성하는 것은 중요하다. 본 절에서는 문



〈그림 2〉 한국 과학 재단의 시스템 구조

서의 의미 전달을 위한 방안으로 아이디를 지정한 방법과 스타일을 이용한 방법으로 제안하였다.

가. 아이디 이용

호글에서는 HWP문서의 내부적인 처리를 위해 누름틀이나 표와 같이 일정한 틀 안에 고유한 아이디를 지정할 수 있는 기능을 제공한다. Ctrl+n, d를 통해 〈그림 4〉와 같이 사용자가 아이디 대화 상자를 이용하여 임의적으로 지정할 수 있으며 형식에 반영되어 처리된다. 아이디를 이용한 서식 설정은 문서의 내용틀이 정해진 표준 문서의 서문, 목차 등에 누름틀을 지정하여 사용자로 하여금 문서를 작성하게 하는 것에 용이하다.

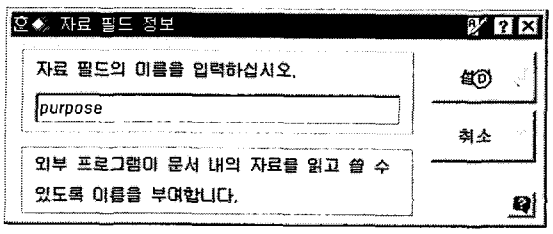
나. 스타일 이용

스타일은 사용자가 자주 사용하는 글자, 문단, 표 등의 모양 정보 등을 정의함으로서 논문, 보고서 등 표준화된 문서의 작성 시 일관성 있는 문단 모양을 제공하여 편집 작업하는데 용이하다. 스타일에 대한 정보는 〈그림 3〉에서와 같이 문서에 정의된 스타일마다 고유한 ID를 제공하여 이를 이용하여 작성한 문서에 대한 서식 정보를 제공한다. 본 논문에서는 정보통신단체표준(TTA표준)에서 정의한 문서 작성 요령을 기준으로 XML 문서 생성을 위해 다음 〈표 1〉과 같은 스타일을 정의하였다.

```
<!DOCTYPE HWPML SYSTEM []>
<HWPML VER = "HWPML 1.1" CODE = KS>
<HEAD>
<INFORMATION>
....
<TITLE> 표준 문서 관리 시스템을 위한 예제 </TITLE>
....
<FIELDDATA TYPE = 4 POSTYPE = 1 ID "purpose">
...
<TEXT><CHAR SIZE = 300><FONTID KOR = 3 ENG = 4> 공공기관에서 통용되는 공 문서
포멧의 표준화를 달성하고, 각 기관간의 공문서.....</FONTID></CHAR></TEXT>
.
<STYLE ID = 36 NAME = "sentences">
<CSTYLE SIZE = 275 ATTR = 6>
<FONTID KOR = 1 ENG = 1 HANJA = 1 JAPAN = 1 ETC = 1></FONTID>
</CSTYLE>
.....

<P STYLE ID = 36><TEXT> 본 표준은 공공기관 간 교환 대상 문서의 분석을 통해 공문서의
유형을 분류하고 그 유형별 문서형 정의를 제안한다 </TEXT>
```

〈그림 3〉 HWPML파일 형식의 예



〈그림 4〉 아이디 지정 대화 상자

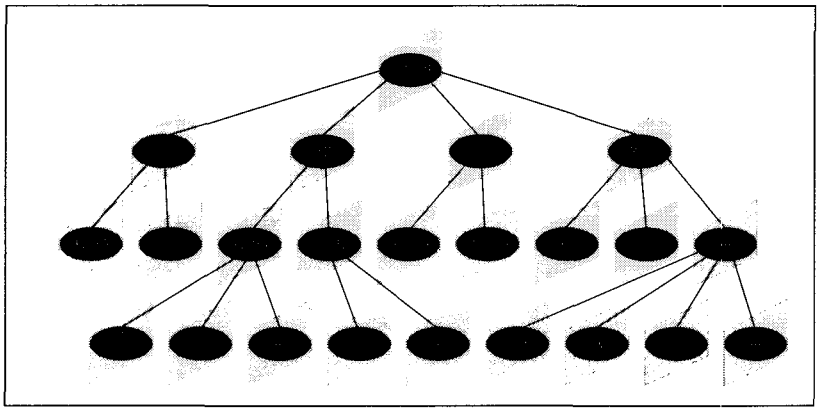
3. 2 DTD 정의

특정 단체의 표준 문서 작성 규정에 따라 정해진 서식을 통해 작성한 문서는 어떠한 형태의 내용이든 XML문서로 생성되어야만하며 이를 위한 DTD의 설계는 매우 중요하다. 본 논문에서는 표준 문서에 대해 제목, 서문, 목차, 본문, 부속서, 부록 등의 서브 엘리먼트로 구성하였으며 각 엘리먼트에 대해 다양한 형태의 문서 구조를 제공하도록 설계하였다.

〈그림 6〉는 설정한 서식 정보와 〈표 1〉에서 정의한 스타일 정보를 기반으로 하여 작성한

〈표 1〉 정보 통산 단체 표준 문서 스타일

스타일 이름	사용 부분	예
chap_n	최상위 장(chapter) 번호	1. 최상위 chapter
chap_n.n	다음 depth의 장 번호	1.1 다음 depth 장
chap_n.n.n	최하위 장 번호	1.1.1 최하위 chapter
list_가	세부 항목 리스트(최상위)	가. 세부 항목 1
list_(1)	세부 항목 리스트	(1) 세부 항목 2
list_(가)	세부 항목 리스트	(가) 세부 항목 3
list_1)	세부 항목 리스트	1) 세부 항목 4
list_가)	세부 항목 리스트	가) 세부 항목 5
list_add	추가 항목 리스트	- 추가 세부 항목
sentences	문장	내용이 들어가는 부분
picture_title	그림 캡션	〈그림 1〉 그림 캡션
table_title	표 캡션	〈표 1〉 표 캡션
annex_title	부속서 제목	부속서 A
appendix_title	부록 제목	부록 I
description_title	'개요 서술' 부분의 제목	개요
schema_title	'표준의 구성' 부분의 제목	구성



〈그림 5〉 정보통신단체표준문서 DTD 다이어그램

```
<!ENTITY % _preface "(purpose, reffered, international, intellectual_right, certification, history*) ">
<!ENTITY % _date "(year, month, day)">

<!ELEMENT standard_doc (doc_title, preface, toc, body, annex?, appendix?)>

<!ELEMENT doc_title (standard_org, standard_no, established_date , korean, english)>
<!ELEMENT standard_org (#PCDATA)>
<!ELEMENT standard_no (#PCDATA)>
.....
<!ELEMENT preface ( ko_preface, en_preface)>
<!ELEMENT ko_preface (%_preface;)>
<!ELEMENT en_preface (%_preface;)>
.....
<!ELEMENT body (chapter+)>

<!ELEMENT chapter (chap_title, ((sentences | pictures | tables | list) | sub_chap1)*)>
....
....
<!ELEMENT lists ( ol_1)>
<!ELEMENT ol_1 (list_title, ((sentences | pictures | tables)* | ol_2)*)>
.....

<!ELEMENT annex (title, chapter+)>

<!ELEMENT appendix (title, chapter+)>
```

〈그림 6〉 정보통신단체표준문서 DTD

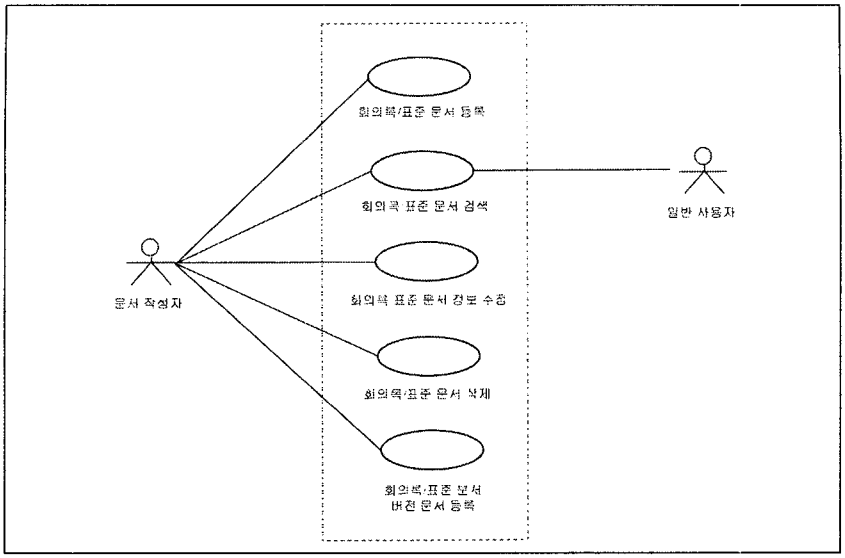
DTD로서 정보통신단체표준문서 작성을 위해 필요한 문서 구조 정보를 정의하였다. 문서의 표지 정보, 서문, 요약문, 목차 등의 정보는 설정한 서식 정보를 기반으로 하였으며 기타 본문, 부록 등의 내용은 정의한 스타일 정보를 기반으로 정의하였다.

4. 표준 문서 관리 시스템의 설계

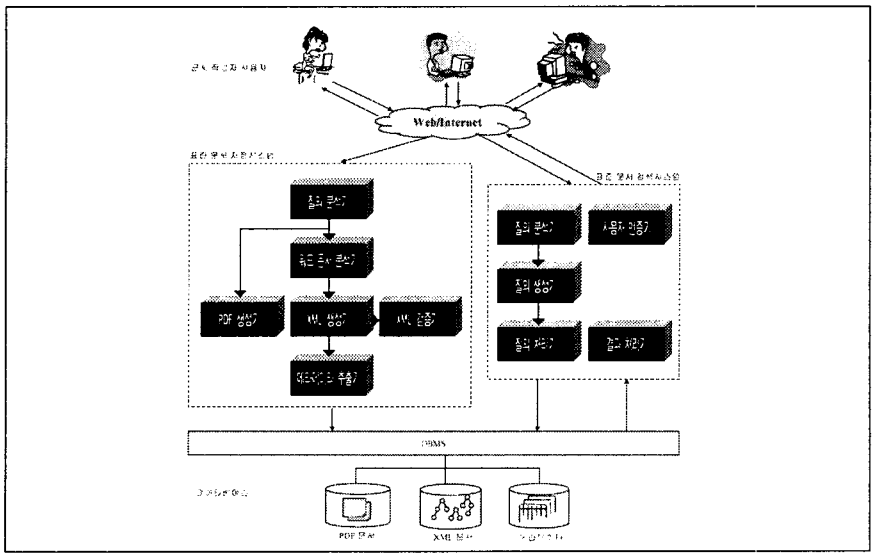
본 논문에서 제안하는 표준 문서 관리 시스템은 문서 작성자가 작성한 문서를 등록하는 단계에서부터 수정, 삭제, 검색 및 표준 문서에 대한 버전 관리 단계까지를 범위로 설계하였다. 〈그림 7〉과 같이 표준 문서 관리 시스템은 문서의 작성, 수정, 삭제 등의 저작권을 가진 문서 작성자와 문서의 검색 권한만을 가진 일반 사용자를 대상으로 제안하였다.

다음의 Use-Case diagram을 기반으로 전체 시스템 구조는 〈그림 8〉와 같이 3개의 계층 즉, 사용자, 표준 문서 관리 시스템 그리고 데이터베이스 구조를 가진다. 상위계층은 문서의 등록, 수정, 삭제를 수행하는 문서 작성자와 검색을 주요 기능으로 수행하는 일반사용자로 구성된다. 상위 계층을 통해 입력된 정보는 표준 문서 관리 시스템을 통해 처리된 후 데이터베이스에 저장, 관리된다.

본 논문에서 설계한 표준 문서 관리 시스템은 표준 문서를 저장하는 부분과 검색하는 부분으로 나누어 설계하였다. 문서의 저장은 문서 등록자에게 입력받은 문서 관련 정보와 문서 파일 정보를 질의 분석기를 통해 분석한 후, 워드 문서 분석기를 통해 XML문서 생성을 위해 프로그램 내에서 처리할 수 있는 HWPML 또는 RTF형태의 파일로 변환한다. 변환된 문서는 XML 생성기를 통해 DTD를 기반으로 XML 문서를 생성되며 메타데이타 추출기를 이용하여 데이터베이스에 저장 할



〈그림 7〉 표준 문서 관리를 위한 Use-case diagram



〈그림 8〉 전체 시스템 구조

정보를 추출한 후 스키마에 맞게 저장한다. 문서의 검색은 사용자가 입력한 질의 조건 및 키워드 등을 질의 분석기를 통해 분석한 후, 질의 생성기를 통해 알맞은 질의어를 생성한다. 생성된 질의어는 질의 처리기를 통해 질의를 수행하며 수행된 결과는 결과 처리기를 통해 사용자에게 알맞은 형태로 보여주게 된다. 또한, 사용자 인증기를 제공하여 편집 가능한 워드 문서에 대해서의 접근 제한 기능을 제공하였다.

4. 1 표준 문서의 등록

표준 문서의 등록은 문서 작성자가 정의된 서식에 따라 작성한 문서를 등록 인터페이스를 통해 upload하여 처리한 후, XML 문서를 생성하여 데이터베이스에 저장하는 기능을 수행한다. 〈그림 9〉은 등록 된 문서의 처리 과정을 Sequence diagram으로 나타낸 것이며, 이러한 처리 과정을 크게 4가지로 나누어 이루어진다.

가. 문서의 upload(회의록/표준 문서 등록 인터페이스)

문서작성자는 등록 인터페이스를 통해 작성한 HWP나 MS Word형태의 워드 프로세서 파일을 upload하며 문서에 관련된 추가 정보를 입력한다.

나. XML 문서 생성(HWP2HML, HML2XML)

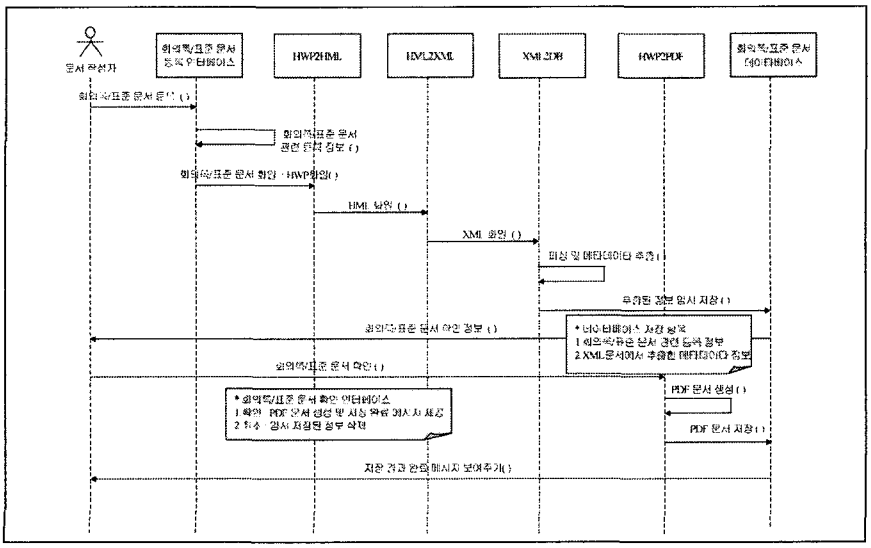
등록 인터페이스를 통해 upload된 파일을 분석하여 XML문서를 생성하는 기능을 제공하며 HWP2HML 모듈과 HML2XML 모듈을 통해 처리된다. HWP2HML모듈은 upload된 워드 프로세서 문서를 프로그램에서 처리할 수 있는 형태인 RTF파일이나 HWPML파일로 변환한다. 변환된 RTF파일이나 HWPML 파일은 HML2XML모듈을 통해 문서 내에 정의된 서식에 대한 정보를 분석한 후 DTD에 맞는 XML문서를 생성한다.

다. 메타데이타 추출 및 저장(XML2DB)

생성된 XML문서는 파서를 통해 파싱한 후 메타데이타를 추출하여 문서 작성자가 문서 등록 시 입력한 추가 정보와 함께 데이터베이스에 임시 저장한다. 저장된 정보는 등록 인터페이스를 통해 등록자에게 확인된다.

라. PDF 문서 생성(HWP2PDF)

PDF문서는 등록자가 등록한 워드 프로세서



〈그림 9〉 표준 문서의 등록에 관한 Sequence diagram

문서를 등록·인터페이스를 통해 확인할 경우 제공되는 기능으로 등록한 HWP나 MS Word 형태의 워드 프로세서 문서를 PDF형태의 문서로 생성한 후 저장하는 기능을 제공한다.

4. 2 표준 문서의 검색

표준 문서의 검색은 전체 문서 검색과 상세 정보 검색으로 제공된다. 일반 사용자는 검색 인터페이스를 통해 데이터베이스에 저장된 표준 문서의 전체 목록 정보를 검색한다. 이들 정보들에 대해 조건 키워드 입력에 의한 문서 검색 및 각 문서에 대한 상세 정보 검색 기능을 제공한다. 또한, 검색 문서 파일에 대한 사용자 인증을 요구하여 문서 저작에 대한 보호 기능을 제공한다.

4. 3 표준 문서의 수정/삭제/버전 관리

가. 표준 문서의 수정

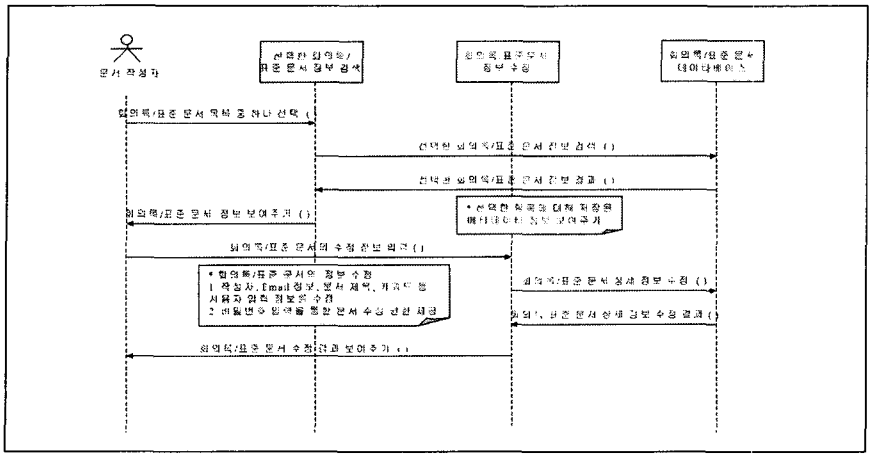
문서 정보의 수정은 문서 파일에 관한 수정과 문서에 관한 입력 정보에 대한 수정으로 나뉘어진다. 본 논문에서는 문서 등록 시 문서 작성자가 입력한 정보의 수정만을 범위로 하였으며 문서 파일에 관한 수정은 등록한 문서를 삭제 한 후, 다시 재등록하는 방안을 제공한다.

나. 표준 문서의 삭제

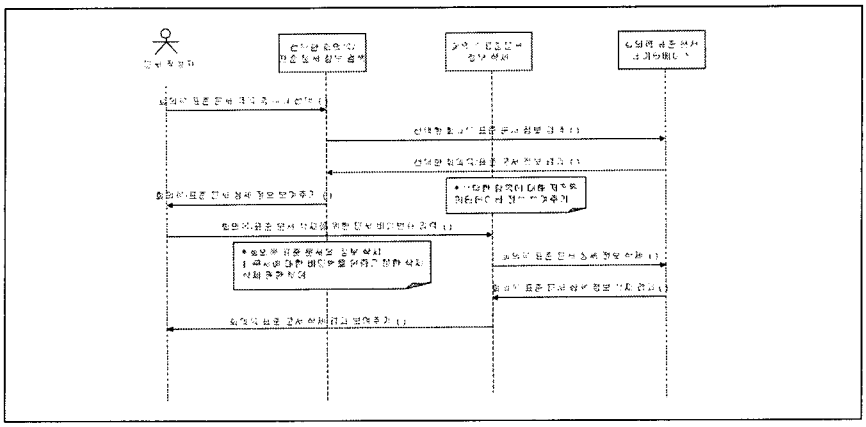
문서 정보의 삭제는 특정 문서에 대해 문서 등록자가 입력한 비밀번호 인증을 통해 이루어진다. 삭제된 문서는 삭제 후 삭제 결과 메시지를 통해 사용자에게 확인한다.

다. 표준 문서의 버전 관리

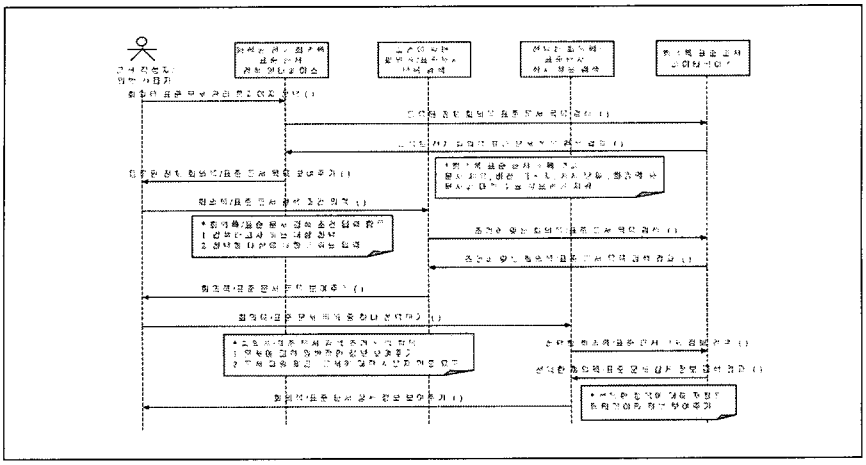
표준 문서의 버전 정보는 프로그램 내에서 자동화 처리되며 사용자는 버전이 수정된 워드 프로세서 문서를 문서 등록과 같이 문서에 관한 추가 정보 입력 및 파일을 upload하여 등록한다. 등록된 문서는 4.1절의 표준 문서 등록의 절차와 동일하게 수행되어 저장, 관리되어 사용자에게 보여진다.



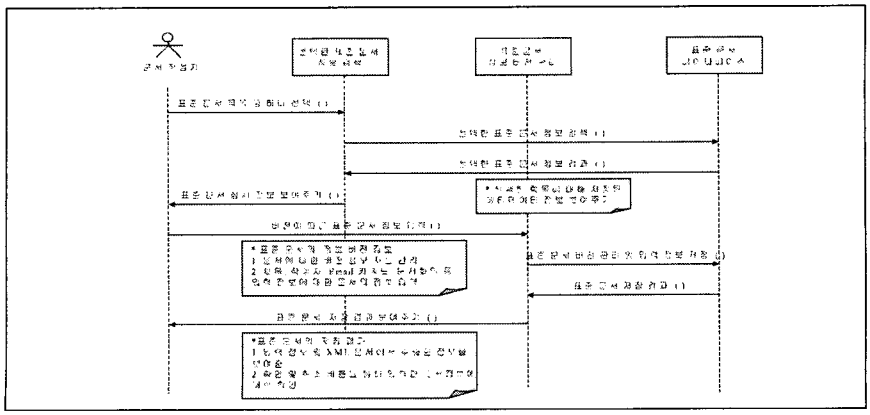
〈그림 10〉 표준 문서의 검색 Sequence diagram



〈그림 11〉 표준 문서의 수정 Sequence diagram



〈그림 12〉 표준 문서의 삭제 Sequence diagram



〈그림 13〉 표준 문서의 버전 관리 Sequence diagram

5. 표준 문서 관리 시스템의 구현

5. 1 구현 환경

본 논문은 다음 〈그림 14〉과 같이 Window NT Server 4.0 플랫폼을 기반으로 C++ 언어를 이용하여 표준 문서 관리 시스템을 구현하였으며, 한글과 컴퓨터에서 제공하는 HWP 워드 프로세서를 기반으로 구성하였다. 표준 문서 저장 시스템은 C++ 언어 외에 생성된 XML 문서를 검증하고 데이터베이스에 저장 할 메타데이타를 추출하기 위해 IBM에서 제공하는 XML4C 파서를 이용하였으며, PDF 문서 생성을 위해 HANQ2PDF툴을 이용하였다. 이 시스템의 데이터를 관리하는 데이터베이스는 SQL Server 7.0을 기반으로 구성하였으며, 사용자는 ASP 3.0을 기반으로 구성되어 있는 제공되는 웹 브라우저를 통해 정보를 입력받거나 제공하도록 하였다.

5. 2 스키마 구성

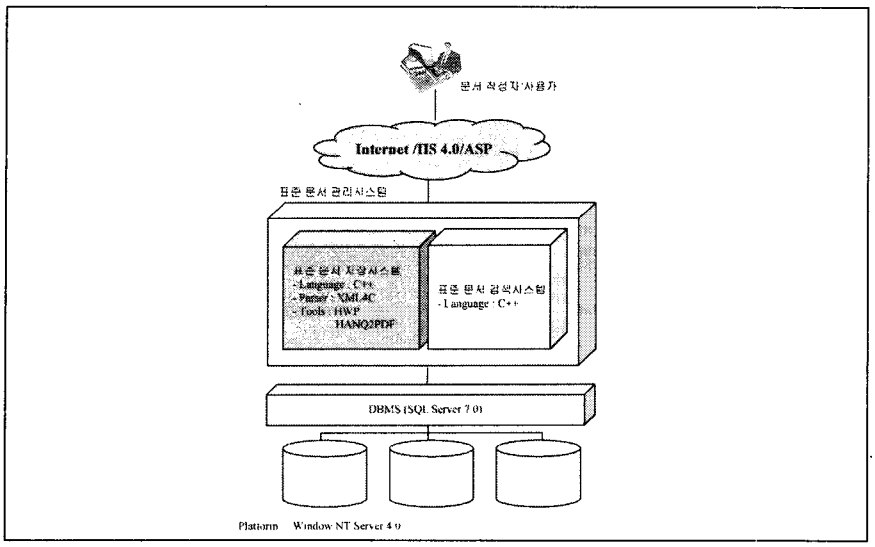
본 논문에서 제공하는 스키마는 크게 4가지의 정보로 구성되어 있다.

가. 문서를 구분하기 위한 id 정보

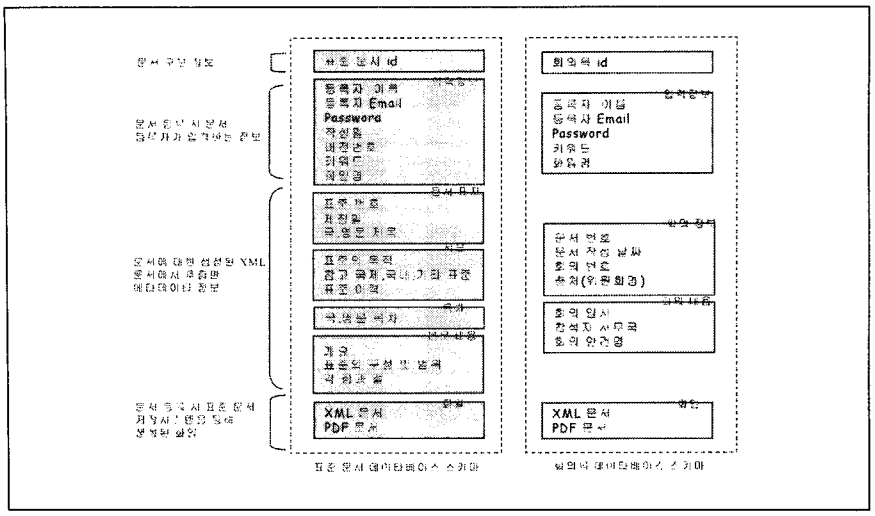
등록 된 문서에 대해 고유한 id를 부여하여 문서의 저장 및 관리한다.

나. 문서 등록 시 문서등록자가 입력하는 정보

표준 문서나 회의록 등 작성한 문서 파일에서 알 수 없는 등록자 이름, 등록자 Email, 등록 패스워드, 문서의 주요 키워드 정보 등 문서 관련 정보를 문서 등록 시 입력받아 관리한다. 입력된 대부분의 정보는 사용자가 문서에 대한 전체 목록 검색 시 간략 정보로 제공된다.



〈그림 14〉 표준 문서 관리 시스템의 구현 환경



〈그림 15〉 표준 문서와 회의록의 스키마

다. 등록 문서에 대한 XML문서에서 추출한 메타데이타 정보

작성한 문서의 내용 중 사용자의 검색에 자주 사용되는 서문, 목차, 요약문 등의 정보를 생성된 XML 문서에서 추출하여 관리한다. 이러한 정보는 사용자에게 문서에 대한 상세 정보로 제공된다.

라. 등록 문서에 대해 생성된 문서 파일에 대한 정보

문서 등록자가 등록한 HWP문서 파일 및 등록으로 인해 자동 생성되는 XML 문서 파일과 PDF 문서 파일을 관리한다. 저장된 문서 파일은 사용자에게 문서 파일 검색 시 인증과 함께 제공된다.

5. 3 표준 문서의 등록 시스템 구현

표준 문서의 등록과 버전 관리 시스템은 문서에 대한 버전 정보 관리를 제외하고는 등록되는 문서 파일에 대해 동일한 루틴을 가지고 저장, 관리된다. 버전 관리 시스템의 경우 동일류의 문서에 대해 최근에 저장된 문서의 버전 정보를 참고하여 제공되도록 구현하였다. 다음은 문서 파일 변환에서 저장까지의 시스템 구현을 설명한 것이다.

가. 파일 변환 시스템 구현

1) HML 또는 PDF 문서 생성

한글에서는 외부 프로그램에서 한글의 기능을 제어하기 위해 DDE API를 제공하며 일반적으로 한글은 DDE 서버, 외부 프로그램은 DDE 클라이언트로 동작한다. DDE API는 작성된 문서 내용을 다른 포맷 형태로 변환, 문서의 인쇄, 새로운 외부 제어용 툴바 등록, 내용 삽입 등 다양하게 문서를 제어할 수 있다. 본 시스템에서는 DDE API를 이용하여 HML 형태의 문서 포맷으로 변환하며 HANQ2PDF를 이용한 HWP 문서 파일의 인쇄를 제공하여 PDF 문서 파일을 생성하였다[5].

가. HWP2HML 문서 생성

HWP2HML 변환기는 HWP 형태의 문서를 HML 형태의 문서로 변환한다. HWP 파일에서 HML 파일로의 실제적인 변환 작업은 한글에서 제공하는 DDE API를 이용하여 구현하였는데, 여기에서 사용되는 주요 프로토콜은 DDE(Dynamic Data Exchange)로서 한글(hwpw.exe)과 HWP2HML(hwpCOM.dll)은 각각 DDE 서버/클라이언트로서의 역할을 하면서 동작하여 실제적인 변환 작업을 수행하도록 구현하였다. HWP 문서를 HML 문서로 변환하는 작업은 〈그림 16〉과 같이 clsConvert 클래스의 ToHML 메소드를 호출함으로써 수행된다. ToHML 메소드에는 두개의 입력 인자를 지정할 수 있도록 되어 있는데, 첫 번째 인자에는 HWP 파일이 위치한 절대패스를 포함한 HWP 파일명을 지정하고, 두 번째 인자에는 변환된 HML 파일이 저장되기를 원하는 위치의 절대패스를 포함한 HML 파일명을 지정하면 된다.

```
// HML 변환을 위한 컴포넌트의 인스턴스 생성
Set obj = CreateObject("hwpConvert.clsConvert")

HMLNameOnly = num & ".hml"
PDFNameOnly = num & ".pdf"

// HWP 파일명(절대패스 포함) 지정
hmlFile = HMLDir & "₩" & HMLNameOnly
// HML 파일명(절대패스 포함) 지정
pdfFile = PDFDir & "₩" & PDFNameOnly

// hmlFile위치에 변환된 HML 파일 저장
lResult = obj.ToHML(hwpFile, hmlFile)
```

〈그림 16〉 HWP2HML 변환을 위한 메소드 호출

```
long CHwp::ToHML(BSTR FAR* pBstrFile, BSTR FAR* pBstrSaveFile)
{
  _bstr_t bstrFile(*pBstrFile);
  WideCharToMultiByte(CP_ACP, 0, bstrFile, -1, strFileName.GetBuffer(512), 512, NULL, NULL);

  ret = LoadHwpFile(strFileName);
  if (ret.errcode == HAPIERR_NO_ERROR) {
     _bstr_t bstrSaveFile(*pBstrSaveFile);
     WideCharToMultiByte(CP_ACP, 0, bstrSaveFile, -1, strSaveFileName.GetBuffer(512), 512, NULL, NULL);

     strType = "HML/KS";
     cmd.Format("SaveDocument(%d,₩"%s₩",₩"%s₩",,)", ret.value, strType, strSaveFileName);
     m_HwpDdeContext.ExecuteHwpCmd(cmd);

     HAPIRET ret;
     m_HwpDdeContext.GetRetVal(&ret);
  }
  return ret.errcode;
}
```

〈그림 17〉 HWP2HML 변환 처리 루틴

나. PDF 문서 생성

HWP2PDF 변환기는 HWP 형태의 문서를 PDF 형태의 문서로 변환한다. HWP 파일에서 PDF 파일로의 실제적인 변환 작업은 HWP 2HML 변환기에서와 같이 한글에서 제공하는 DDE API를 이용하여 구현하였는데, 여기에서 사용되는 주요 프로토콜은 DDE(Dynamic Data Exchange)로서 한글(hwpw.exe)과 HWP2HML(hwpCOM.dll)은 각각 DDE 서버/클라이언트로서의 역할을 하면서 동작하여 실제적인 변환 작업을 수행하도록 구현하였다.

하지만, HWP 문서를 PDF 문서로 변환하는 작업은 HWP2HML 변환기와는 달리 웹 기반이 아닌 hwpMkPDF.exe라는 독립 실행 프로그램에 의해 만들어 진 인터페이스에 의해 수행되도록 구현하였다. 이는 실질적으로 PDF 파일을 생성하는 PDF 생성 프로그램(한 QPDF1.0에서 제공)과 'HWP프로그램'이 서로 독립적인 처리 루틴을 갖고 수행되고 PDF 생성 프로그램은 PDF 파일을 저장하기 위해 사용자의 결정이 필요한 과정, 즉 저장할 PDF 파일명을 지정하기 위한 대화상자를 필요로 하기 때문이다. 따라서 hwpMkPDF 프로그램은 관리자의 제어하에 등록된 문서에 대한 PDF 생성 작업을 수행하도록 구현하였다.

hwpCOM 클래스의 ToPDF 메소드에는 두개의 입력 인자를 지정할 수 있도록 되어 있는데, 첫번째 인자에는 HWP 파일이 위치한 절대패스를 포함한 HWP 파일명을 지정하고, 두번째 인자에는 변환된 PDF 파일이 저장되기를 원하는 위치의 절대패스를 포함한 PDF 파일명을 지정하면 된다. hwpCOM 클래스의 ToPDF 메소드는 변환 작업에 성공하면 0값을 실패하면 오류 번호를 리턴한다.

hwpCOM 클래스의 ToPDF 메소드는 HWP 포맷의 파일을 PDF 포맷의 파일로 변환하기 위하여 Component화 되어 수행되는 모듈로서, 'HWP프로그램'과 DDE 통신한다. hwpCOM 클래스의 ToPDF 메소드는 〈그림 18〉와 같이, 한글에서 제공하는 Print-

```
long CHwp::ToPDF(BSTR FAR* pBstrFile, BSTR FAR* pBstrSaveFile)
{
  _bstr_t bstrFile(*pBstrFile);
  WideCharToMultiByte(CP_ACP, 0, bstrFile, -1, strFileName.GetBuffer(512), 512, NULL, NULL);

  ret = LoadHwpFile(strFileName);
  if (ret.errcode == HAPIERR_NO_ERROR) {
     _bstr_t bstrSaveFile(*pBstrSaveFile);
     WideCharToMultiByte(CP_ACP, 0, bstrSaveFile, -1, strSaveFileName.GetBuffer(512), 512, NULL, NULL);

     cmd.Format("PrintDocument(%d,)", ret.value);
     m_HwpDdeContext.ExecuteHwpCmd(cmd);

     HAPIRET ret;
     m_HwpDdeContext.GetRetVal(&ret);
  }
  cmd.Format("CloseDocument(%d,)", ret.value);
  m_HwpDdeContext.ExecuteHwpCmd(cmd);

  if (eFlag == 100)        return ret.errcode;
  else                     return eFlag;
}
```

〈그림 18〉 HWP 및 PDF 문서 변환

Document 함수를 호출하여 'HWP프로그램'에게 PDF 파일로 HWP 파일을 변환하고 저장하도록 요구하며 'HWP프로그램'은 지정된 HWP 파일을 PDF 파일로 변환 저장하라는 요구를 PDF 생성 프로그램에게 전달한다. 최종적으로 PDF 생성프로그램은 요구된 HWP 파일을 PDF 파일로 변환, 저장하는 기능을 수행한다

2) XML 문서 생성

가. HML2XML

등록 시스템을 통해 접수된 HWP 파일을 DDE API를 통하여 변환한 HML 파일은 마크업(markup) 형태로 이뤄져서 외부 프로그램을 통한 데이터 추출이 가능하다. 이런 특성을 이용하여 HML 파일에서 특정 내용을 추출하여 XML 문서를 만들게 된다.

■ HML 파일의 분석

HML 형태인 파일을 XML 형태로 변환하기 위하여, 먼저 HML 파일 내에서 어떤 태그가 쓰였으며 그 태그들이 무엇을 의미하는지를 분석해야 한다. 또한 XML 파일을 생성할 때 기본적인 XML 문서의 형식을 따라야 하며, 미리 작성된 DTD에 부합 되야 한다는 점도 중요하다.

HML 파일은 일반적으로 〈그림 19〉과 같은 구조를 지닌다. 〈!DOCTYPE HWPML SYSTEM [ ]〉는 이 문서가 HML 형식임을 나타내주게 되며, 〈HWPML VER="HWPML1.1" CODE=KS〉는 버전 정보 등을 나타내고, 〈HEAD〉 부분엔 현재 HML 문서의 일반적인 정보 등과 지정된 스타일의 번호를 부여하는 등의 내용을 담고 있다. 〈HEAD〉 부분이 끝나고 나면 〈BODY〉 태그가 나타나고

```
<!DOCTYPE HWPML SYSTEM []>
<HWPML VER="HWPML1.1" CODE=KS>
<HEAD>
...
<STYLE ID=2 NAME="appendix_title">
...
</HEAD>
<BODY>
...
<CTRLCODE ID=5>
<FIELDDATA TYPE=4 POSTYPE=1 ID="standard_org">
<FIELDRESULT>표준 단체</FIELDRESULT>
...
</CTRLCODE>
<TEXT>정보통신연구소</TEXT>
<CTRLCODE ID=5>
<FIELDDATA TYPE=4>
</CTRLCODE>
...
<P STYLEID=2>
...
<TEXT>부록 가. 분산 토큰 제어</TEXT>
...
</BODY>
```

〈그림 19〉 일반적인 HWPML 파일

이 태그 아래에 나오는 부분이 HWP 문서의 본문이 HML 형식으로 바뀐 것이다.

〈BODY〉 태그 안에서 〈CTRLCODE ID=5〉〈FIELDDATA TYPE=4 POSTYPE=1 부분은 HWP 문서에서 누름틀, 아이디를 나타낸다. 그리고 그 태그 안의 ID="standard_org" 부분은 서식 중에 아이디로 지정된 부분을, 〈P STYLEID=2〉 부분은 서식 중에 스타일로 지정된 부분을 나타낸다. 〈TEXT〉와 〈/TEXT〉 사이의 내용은 HWP 문서 내에서 실제 문자들을 나타내게 되므로, HML의 이런 형식을 분석하여 데이터들을 추출할 수 있게 되었다.

■ HML 파일의 내용 추출

HML 형식의 특성을 분석하면 해당되는 파일의 내용을 추출할 수 있다. 문서에서 "표준 관련 기관"에 대한 내용을 추출하는 예를 들면, 〈그림 19〉에서 ID="standard_org" 부분이 "표준 관련 기관"에 해당하는 내용을 아이디로 지정한 부분이기 때문에, ID="standard_org" 이후의 내용 중에 〈TEXT〉 태그와 〈/TEXT〉 태그 사이의 내용 중에 다른 태그가 포함되어 있으면 그 부분을 제외한 문자열 부분만을 추출하면 된다. 유의할 점은 작성자가 해당 문자열에 엔터를 입력한 경우엔 〈TEXT〉 태그의 쌍이 반복되므로, 데이터 추출 과정에서 〈CTRLCODE ID=5〉 태그를 데이터 추출의 마침으로 인식해야 한다.

또한, 스타일로 지정된 부분의 데이터를 추출하기 위한 방법은 아이디를 사용하는 방법과 거의 유사하다. 단지, 스타일을 지정했을 경우엔 아이디와 다르게 본문에선 지정된 STYLE ID를 숫자로 가지고 있기 때문에, "부록"에 관한 내용을 추출한다면, 〈그림 19〉에서 〈P STYLEID=2〉 이후의 〈TEXT〉 태그의 쌍을 조사해야 한다. 여기서 STYLE ID는 문서의 〈HEAD〉 부분 안에 〈STYLE ID=2 NAME="appendix_title"〉과 같이 정의되어 있는 경우이다.

아이디와 스타일을 가지고 위의 방법대로 데이터 추출을 할 수 있으며, 데이터 추출에 있어서 처리 시간을 줄이기 위하여, 추출할 부분이 포함되어 있지 않은 〈HEAD〉 부분은 일정 줄(line)을 생략하고 추출할 부분이 시작되는 〈BODY〉 부분 근처에서 데이터 추출을 시작하도록 한다.

■ XML 파일 생성

파일 변환 시스템에서는 데이터 추출과 동시에 하나의 XML 파일을 생성하게 된다. XML 파일의 일반적인 형식을 따르고 사용되는 DTD를 명시하는 부분을 포함하기 위하여 〈그림 20〉 부분은 기본적으로 생성된 파일에 쓰여지게 된다.

〈그림 20〉의 내용에 HML에서 추출된 문자열을 삽입하는 과정에서 XML의 태그를 붙여

```
<?xml version="1.0" encoding="EUC-KR"?>
<!DOCTYPE standard_doc SYSTEM "standard_doc.dtd">
```

〈그림 20〉 생성된 XML의 기본 형식

주는 과정이 있어야 한다. 예를 들면, 정보통신 단체표준문서 DTD에 따르면, 표준 관련 기관을 나타내는 〈standard_org〉는 루트 엘리먼트인 〈standard_doc〉의 하위 엘리먼트인 〈doc_title〉의 하위 엘리먼트로 존재해야 한다. 그러므로, 〈그림 20〉의 XML 문서에 〈standard_doc〉와 〈doc_title〉이라는 태그를 삽입하고, 〈standard_org〉라는 태그도 삽입한 후에 표준 관련 기관의 이름을 이전에 설명한 방법으로 추출하여 삽입하고, 〈/standard_org〉를 삽입하면, 〈그림 21〉과 같은 형태가 된다.

나. 메타데이타 추출 시스템 구현

HML, PDF, XML 문서 생성기를 통해 생성된 정보를 사용자의 입력 정보와 함께 데이터베이스에 저장한다. 메타데이타 추출 시스템은 생성된 XML 문서 파일을 읽어 들여 문서의 메타 데이터 정보를 추출하여 정의된 데이터베이스 스키마에 맞도록 저장하는 기능을 제공한다. 데이터베이스 스키마는 〈그림 15〉에서 언급하였듯이 사용자가 문서를 웹 인터페이스를 통해 등록할 때 입력한 정보, 생성된 XML 문서의 엘리먼트 값에서 얻은 정보, 그리고 생성 파일 정보들로 구성되어 있다. 이 정보 중 메타데이타 추출 시스템은 데이터베이스에 저장된 XML 문서 파일을 읽어 XML4C 파서를 이용하여 사용자가 등록한 워드 문서에서 추출할 수 없는 문서 내용을 XML 문서를 이용하여 추출하여 저장한다. 〈표 2〉은 정의된 데이터베이스 스키마와 DTD 간의 매핑 관계를 나타낸 것으로 것을 보여준다.

메타데이타 추출 시스템은 크게 두 가지의 기능 구현으로 나누어질 수 있다. 생성된 XML 문서 파일명을 읽어 XML4C 파서를 이용하여 원하는 정보를 추출하는 부분과 추출된 정보들을 데이터베이스 스키마에 맞도록 저장하는 부분으로 나뉘어진다. 〈그림 22〉는 이를 위한 간략한 프로그램을 정의한 것이다.

다. 인터페이스

표준 문서 관리 시스템의 초기 화면은 일반적으로 사용되는 게시판과 동일하며 사용자 또는 문서 등록자는 표준 문서 관리 시스템 접속 시 현재 데이터베이스에 저장되어 있는 표준 문서 정보의 간략한 정보와 함께 전체 검색 목록을 검색하게 된다. 새로운 문서의 등록은 필 아웃-폼 형식으로 문서의 추가 정보를 입력하며 '찾아보기' 항목을 통해 등록할 문서를 디렉토리에서 찾아 선택하여 등록하도록 하였다.

```
<?xml version="1.0" encoding="EUC-KR"?>
<!DOCTYPE standard_doc SYSTEM "standard_doc.dtd">
<standard_doc>
<doc_title>
<standard_org>정보통신단체</standard_org>
....
```

〈그림 21〉 표준 관련 기관 정보가 삽입된 XML 문서

〈표 2〉 DTD와 SQL Server Schema 매핑 관계

설 명	DTD(Element)	SQL Server Schema
표준 문서 id	없음	id(int)
등록자 이름	없음	name(varchar)
이메일	없음	email(varchar)
비밀번호	없음	passwd(varchar)
주제	없음	subject(varchar)
게시일	없음	inputData(samlldatetime)
파일 크기	없음	filesize(Float)
조회수	없음	cnt(int)
버전 관리를 위한 정보	없음	relplyid(int)
	없음	relplylevel(int)
	없음	relplynum(int)
최신 버전 정보	없음	verInfo(int)
문서의 버전	없음	version(int)
작성일	없음	date(datetime)
키워드	없음	keyword(varchar)
표준 번호	standard_no	standard_no(varchar)
표준 제정 기관	standard_org	standard_org(varchar)
제정일	established_date	established_date(datetime)
국문 제목	korean	ko_title(varchar)
영문 제목	english	en_title(varchar)
표준의 목적	purpose	purpose(varchar)
참고 국제 표준	ab_standard	ab_standard(varchar)
참고 국내 표준	dom_standard	dom_standard(varchar)
참고 기타 표준	other_standard	other_standard(varchar)
표준 이력	history	history(varchar)
국문 목차	ko_toc	ko_toc(varchar)
영문 목차	en_toc	en_toc(varchar)
개요	introduction	introduction(varchar)
표준의 구성 및 범위	scope	scope(varchar)
hwp 파일 이름	없음	hwp(varchar)
xml 파일 이름	없음	xml(varchar)
pdf 파일 이름	없음	pdf(varchar)

5. 4 표준 문서의 검색/수정/삭제 시스템 구현

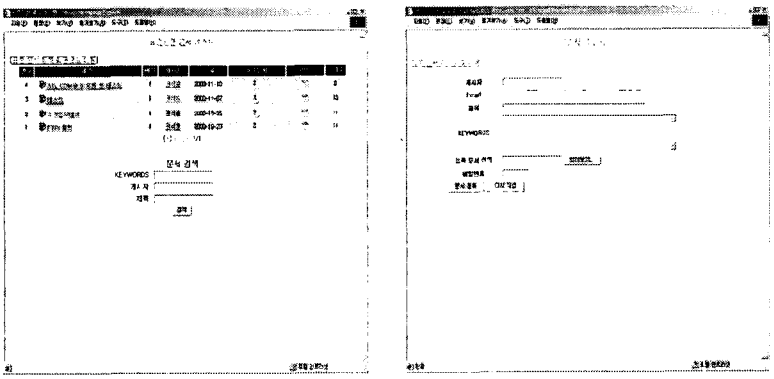
등록을 제외한 검색, 수정, 삭제 및 버전 관리 시스템은 ASP를 이용하여 ODBC를 기반으로 SQL Server와 연동하여 데이터 관리하도록 구현하였다. 〈그림 24〉은 검색 프로그램 중에 ODBC와 연동하여 SQL문을 수행하는 부

```
XML2DB::XML2DB()
{
  // 초기화
                ....
  DOMParser parser;
  Parser.setValidationSchema(valSchema);
  Parser.setDoNamesspaces(doNamespaces);
                ...
  Parser.parse(xmlFile);
                ...
 extract(ifstream fp);
 ...
 store2DB(char *data);
 ..
}

//XML 문서에서 메타데이타를 추출한다.
Void XML2DB::extract( ifstream fp)
{
}

//추출된 메타데이타를 DB에 저장한다.
Void XML2DB::store2DB(char *data)
{
}
```

〈그림 22〉 메타데이타 추출 프로그램



〈그림 23〉 표준 문서 관리 시스템 초기화면 및 등록 화면

```
<%
'Set oConn = Server.CreateObject("ADODB.Connection")
'oConn.Open....
SQL = "SELECT * From data where name LIKE '%"
SQL = SQL & Request.QueryString("poster") & "%' AND subject LIKE '%"
SQL = SQL & Request.QueryString("subject") & "%' AND keyword LIKE '%"
SQL = SQL & Request.QueryString("keyword") & "%"
SQL = SQL & "ORDER BY replyid DESC,"
SQL = SQL & "Version DESC, seq ASC"
%>

'Set oRS = Server.CreateObject("ADODB.RecordSet")
....
```

〈그림 24〉 표준 문서 관리 시스템의 검색 프로그램

분을 기술한 것이다.

6. 결 론

본 논문에서는 표준 문서를 위한 관련 기술로 현재 국내?외에 수행되고 있는 연구 사례를 조사하였으며 현재의 시스템 환경을 기반으로 표준 문서 제정 과정에서 이루어지는 표준 문서 정보 교환을 보다 효과적인 수행하기 위한 표준 문서 관리 시스템을 설계 및 구현하였다.

본 논문에서 구현한 표준 문서 관리 시스템은 기존 시스템 환경의 변화 없이 상호간의 문서 정보를 쉽게 교환하며 관리할 수 있는 기능을 제공하며, 표준 문서 제정 과정 중인 문서에 대한 저작권을 보호하기 위해 일반 사용자로 하여금 편집 불가능한 형태인 PDF 문서 파일로 생성하여 제공하였다.

앞으로의 연구 방향은 등록된 워드 프로세서 문서의 메타데이타뿐만 아니라 문서 전체 내용을 관리하며 XML 문서의 특징을 이용한 다양한 형태의 검색 기능을 제공할 수 있는 XML 전용 데이터베이스를 이용한 시스템을 구현하는 것과 더 확장하여 여러 개로 분산되어 있는 데이터베이스에서 원하는 정보들을 통합하여 검색할 수 있는 시스템을 구현하는 것이다.

참 고 문 헌

[1] Bray, T., et al., "Extensible Markup Language(XML) PR-xml, 971208, W3C, http://www.w3.org/TR, 1997

[2] invisible World Inc, http://www.invisible.net/developers

[3] RFC Editor, http://www.rfc-editor.org/

[4] HWPML(한글 마크업 언어) 매뉴얼 97 기능 강화판, 한글과 컴퓨터.

[5] 한글 EED API 안내, 한글과 컴퓨터.

[6] TTA 단체 표준 문서, 한국 정보 통신 기술 협회, http://www.tta.or.kr

[7] 전병선. 1999. Microsoft Visual C++ 6.0 ATL COM Programming. 삼양출판사

[8] Kate Gregory. 1998. Visual C++ 6.0. 정보문화사.

[9] 오해석, 임승린, 김은영. 2000. ASP를 이용한 웹 사이트 구축. 홍릉과학출판사.

[10] Alex Homer 외 14명 공저. 2000. Professional Active Server Page 3.0. 정보문화사.

[11] 이상엽 저, Visual C++ Programming Bible Ver 6.x, 영진출판사.

[12] Xerces-C API documentation, http://xml.apache.org/xerces-c/api.html

〈부 록〉

```
〈!ENTITY % _preface
        "(purpose, reffered, international, intellectual_right, certification, history*)"〉
〈!ENTITY % _date "(year, month, day)"〉
** 정보 통신 표준 문서의 표지, 서문, 목차, 요약문 **
〈!ELEMENT standard_doc (doc_title, preface, toc, body, annex?, appendix?)〉
〈!ELEMENT doc_title (standard_org, standard_no, established_date, korean, english)〉
〈!ELEMENT standard_org (#PCDATA)〉
〈!ELEMENT standard_no (#PCDATA)〉
〈!ELEMENT established_date (%_date;)〉
〈!ELEMENT year (#PCDATA)〉
〈!ELEMENT month (#PCDATA)〉
〈!ELEMENT day (#PCDATA)〉
〈!ELEMENT korean (#PCDATA)〉
〈!ELEMENT english (#PCDATA)〉
〈!ELEMENT preface (ko_preface, en_preface)〉
〈!ELEMENT ko_preface (%_preface;)〉
〈!ELEMENT en_preface (%_preface;)〉
〈!ELEMENT toc (ko_toc, en_toc)〉
〈!ELEMENT ko_toc (#PCDATA)〉
〈!ELEMENT en_toc (#PCDATA)〉
** 정보 통신 표준 문서의 본문 **
〈!ELEMENT body (chapter+)〉
〈!ELEMENTchapter
(chap_n|sentences|chap_n.n|ol_1|ol_2|ol_3|ol_4|ol_5|picture_title|chap_n.n.n|table_title)*〉
     〈!ATTLIST chapter seq CDATA #REQUIRED〉
〈!ELEMENT chap_n (#PCDATA)〉
〈!ELEMENT chap_n.n (#PCDATA)〉
〈!ELEMENT chap_n.n.n (#PCDATA)〉
〈!ELEMENT sentences (#PCDATA)〉
〈!ELEMENT ol_1 (#PCDATA)〉
〈!ELEMENT ol_2 (#PCDATA)〉
```

<!ELEMENT ol_3 (#PCDATA)>
<!ELEMENT ol_4 (#PCDATA)>
<!ELEMENT ol_5 (#PCDATA)>
<!ELEMENT picture_title (#PCDATA)>
<!ELEMENT table_title (#PCDATA)>
<!ELEMENT picture (#PCDATA)>
<!ELEMENT tables (#PCDATA)>
<!ELEMENT annex (annex_title, chapter+)>
<!ELEMENT annex_title (#PCDATA)>
<!ELEMENT appendix (appendix_title, chapter+)>
<!ELEMENT appendix_title (#PCDATA)>
<!--ENTITY Definition-->
<!ELEMENT purpose (#PCDATA)>
<!ELEMENT reffered (ab_standard, dom_standard, other_standard)>
<!ELEMENT ab_standard (#PCDATA)>
<!ELEMENT dom_standard (#PCDATA)>
<!ELEMENT other_standard (#PCDATA)>
<!ELEMENT international (#PCDATA)>
<!ELEMENT intellectual_right (#PCDATA)>
<!ELEMENT certification (#PCDATA)>
<!ELEMENT history (th_edition, issued_date, issued_contents)>
<!ELEMENT th_edition (#PCDATA)>
<!ELEMENT issued_date (%_date;)>
<!ELEMENT issued_contents (#PCDATA)>