XOMT도와 SGML DTD 데이타베이스 사이의 상호 변환 시스템의 개발†

o
임혜정*, 김성운*, 류은정*, 박인호*, 강현석*
*경상대학교 컴퓨터과학과/전산개발연구소

Development of A Transformation System Between XOMT Diagram and SGML DTD Database†

o
Hye-Jung Lim*, Sung-Un Kim*, Eun-Jung Ryu*, In-Ho Park*, Hyun-Syug Kang*
*Dept. of Computer Science/The Institute of Computer Research and Development
GyeongSang Nat'l Univ.

요 약

최근 사용이 증가하고 있는 SGML 전자 문서들은 객체 지향 데이타베이스를 통해 보다 효과적으로 관리할 수 있다. 이를 위해서는 SGML DTD를 체계적으로 다룰 수 있는 객체 지향 방법론이 요구되는데, 이에 따라 새로운 객체 지향 다이아그래밍 방법으로 XOMT도[3]가 제안된 바 있다.

본 논문은 이러한 XOMT도로 설계한 SGML DTD 문서를 객체 지향 데이타베이스에서 체계적으로 관리할 수 있도록 XOMT도-DB 상호 변환 시스템(TSXD)을 개발한 내용을 기술한다. TSXD는 크게 XOMT도 관리 부시스템(XDMS), XOMT도-DTD 변환 부시스템(XDTS), DTD-XOMT도 변환 부시스템(DXTS)으로 구성되어 있다.

1. 소개

멀티미디어 전자 문서를 기술하는 표준 메타 언어로 SGML이 다양한 응용에 사용되고 있다. 이러한 SGML 전자 문서들을 객체 지향 데이타베이스로 관리하면 효과적이다[1, 2]. 그런데 SGML 전자 문서의 논리적 구조를 기술하는 DTD(Document Type Definition)를 객체 지향 데이타베이스에서 체계적으로 다루기 위해서는 객체 지향 개발 방법론이 정립되어야 하는데, 이를 위해 최근 XOMT도[3]가 제안된 바 있다. XOMT도는 OMT[4]의 객체도를 SGML DTD 기술에 적합하게 변경, 확장한 것으로 특히, SGML DTD 문서들을 객체 지향 데이타베이스에서 관리하기 쉽게 한다.

본 논문은 이러한 작업이 적절하게 수행될 수 있도록 XOMT도로 설계된 SGML DTD를 객체 지향 데이타베이스에 변환하여 저장하고, 반대로 데이타베이스에 저장된 DTD 정보나 새롭게 텍스트 형태로 작성되어 입력된 DTD 정보를 XOMT도로 재변환하여 그림으로 나타내는 시스템인 TSXD(Transformation System Between XOMT Diagram and DTD Database Schema)를 설계하고 구현한 내용을 기술한다. 전자의 기능은 XOMT도-DTD 변환 부시스템(XOMT Diagram-DTD Transformation Subsystem, XDTS)이 수행하며, 후자의 기능은 DTD-XOMT도 변환 부시스템(DTD-XOMT Diagram Transformation Subsystem, DXTS)이 수행한다.

본 논문의 구성은 다음과 같다. 2장에서는 예를 이용하여 XOMT도 기법에 대해 간략히 소개하고, 3장에서는 SGML DTD 정보 관리를 위한 객체 지향 데이타베이스 스키마에 대해 기술한다. 4장에서는

† 본 연구는 정보통신관리단의 '97 대학기초 연구지원 사업의 지원에 의한 것입니다.

본 논문의 주요 내용인 XOMT도와 객체 지향 데이타베이스에 저장된 텍스트 형태의 DTD 사이에 상호 변환하는 시스템에 대해 기술한다. 5장에서는 결론 및 향후 연구 과제에 대해 논한다.

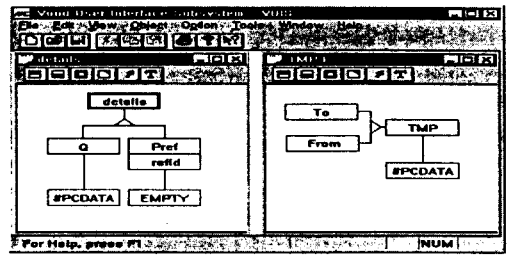
2. DTD 설계를 위한 객체 다이아그래밍 기법

Herwijnen은 SGML DTD 설계를 위한 다이아그래밍 기법으로 구조도를 소개했다[5]. 이 방법은 DTD를 일반적인 트리 구조로 나타낸 것으로 각 노드를 왼쪽에서 오른쪽으로 늘어 놓은 흐름도 형식이다. 그러나 이는 간단한 DTD를 표현하기에는 편리하지만 엔티티라든지 속성리스트를 가지는 엘리먼트의 표현이나 공유되는 부분의 표현 등과 같이 복잡한 DTD를 표현하기에는 어려운 문제가 있다. 따라서 최근 보다 복잡한 DTD도 쉽고 체계적으로 작성할 수 있게 하는 새로운 다이아그래밍 기법으로, 객체 지향 개발 방법론으로 널리 사용하고 있는 OMT의 객체도를 변경하고 확장하여 XOMT(eXtended OMT)가 제시된 바 있다[3].

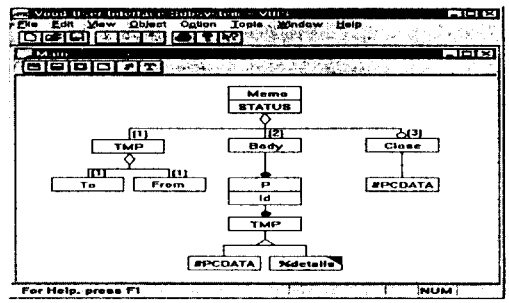
<그림1>은 아래의 일반 메모지 형식에 대한 SGML DTD[5]를 XOMT도로 표현한 것이다.

```
<!-- DTD for simple office memoranda                    -->
<!ENTITY * details   "Q|Pref"                            >

<!--       ELEMENTS   MIN    CONTENT (EXCEPTIONS)        -->
<!ELEMENT  Memo       - -    ((To & From), Body, Close?)>
<!ELEMENT  (To|From)  - O    (#PCDATA)                   >
<!ELEMENT  Body       - O    (P*)                        >
<!ELEMENT  P          - O    (#PCDATA|%details;)*        >
<!ELEMENT  Q          - -    (#PCDATA)                   >
<!ELEMENT  Pref       - O    EMPTY                       >
<!ELEMENT  Close      - O    (#PCDATA)                   >

<!--       ELEMENTS   NAME   VALUE              DEFAULT-->
<!ATTLIST  Memo       STATUS (confiden|public) public  >
<!ATTLIST  P          id     ID                #IMPLIED >
<!ATTLIST  Pref       refid  IDREF             #REQUIRED>
```

<그림 1>에서는 DTD의 주 트리 부분이 그려진 Main 부윈도우, 각각의 Entity마다 하나씩 그려진 Entity 부윈도우, 왼쪽 참조에 대해 그려진 TMP 부윈도우가 있다. 그림에서 각 엘리먼트는 네모 상자로 표현되며 엔티티는 테두리가 이중선으로 된 네모 상자로 나타난다. 엘리먼트의 속성들은 XOMT도에서 엘리먼트 클래스 상자의 두번째 위치에 표시된다. 또한 상·하위 객체간의 is-a 관계성은 삼각형으로, part-of 관계성은 마름모로 나타난다. 발생 지시자인 *(zero or more)는 ●으로, +(one or more)는 +로, ?(optional zero or more)는 ○로 표현된다.



(a) Entity 부윈도우 (b) TMP 부윈도우



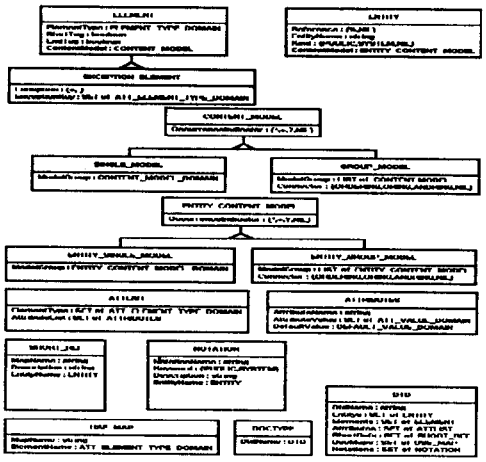
(c) Main 부윈도우

<그림 1> 메모 DTD의 XOMT도

3. DTD 관리를 위한 객체 지향 데이타베이스 스키마

XOMT도로 설계된 SGML DTD를 보다 효율적으로 관리할 수 있도록 객체 지향 데이타베이스 스키마 설계가 요구되며 이를 위해 SGML DTD를 통합적으로 다룰 수 있도록 <그림 2>와 같이 SGML DTD 메타 스키마를 OMT의 객체도로 설계하였다.

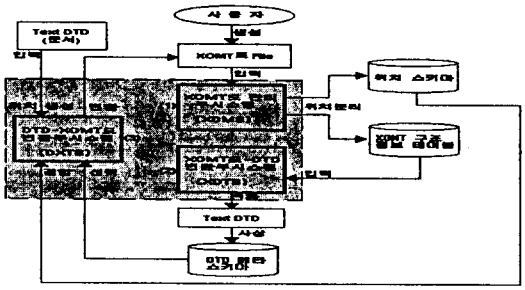
SGML DTD에서 기술될 수 있는 키워드(<!DOCTYPE, <!ENTITY, <!ELEMENT, <!NOTATION, <!ATTLIST, <!SHORTREF, <!USEMAP 중 하나)로 시작하는 구성요소들을 각각 묶어 7개의 기본 클래스들(DOCTYPE, ENTITY, ELEMENT, NOTATION, ATTLIST, SHORTREE, USEMAP)로 정의하고, 키워드와 함께 기술된 구성요소들의 구문에 해당하는 파라미터(parameter)들을 각 클래스의 속성들로 표현하였다. 예를 들어, 클래스 ELEMENT는 엘리먼트 이름부를 ElementType으로, 시작 태그를 StartTag로, 종료 태그를 EndTag로, 내용 모델을 ContentModel로 하는 속성을 가진다.



<그림 2> SGML DTD의 데이타베이스 스키마

4. XOMT도와 데이타베이스 사이의 상호 변환 시스템(TSXD)

XOMT도로 설계된 SGML DTD 정보를 객체 지향 데이타베이스에서 관리하기 위해서는 XOMT도와 DTD 데이타베이스 사이에 상호 변환 시스템(Transformation System Between XOMT Diagram and Database Schema,TSXD)이 개발되어야 한다. TSXD는 <그림 3>과 같이 크게 3개의 부시스템들로 구성되며, 이들은 각각 XOMT도를 관리하는 부시스템(XDMS), XOMT도로 생성된 객체 정보를 DTD 데이타베이스로 변환하는 부시스템(XDTS), 그리고 텍스트 형태나 데이타베이스에 저장된 DTD 정보를 XOMT도로 변환하는 부시스템(DXTS)이다. 이중 XDMS는 XDTS와 DXTS의 교량적 역할을 담당하며, XDTS와 DXTS는 상호 변환을 수행하는 부시스템들이다.



<그림 3> XOMT도-DTD 사이의 상호 변환 시스템

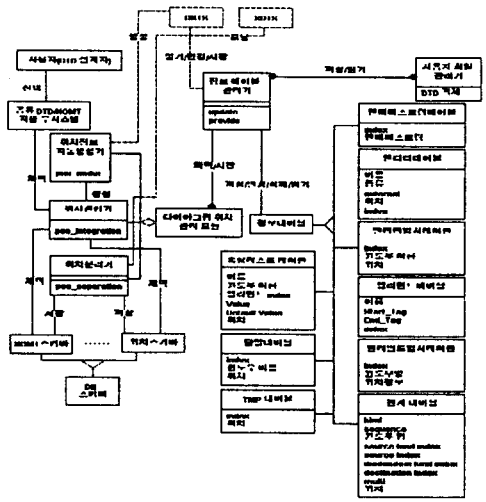
4.1 XOMT도 관리 부시스템(XDMS)

XOMT도 관리 부시스템(XOMT Diagram Management Subsystem, XDMS)은 XOMT도와 객체 지향 데이타베이스 사이에 상호 변환이 효율적으로 이루어질 수 있도록 교량적 역할을 담당한다. 이를 위해 XDMS는 XOMT도를 이용하여 생성, 수정된 SGML DTD 정보 화일을 관리하는 사용자 화일 관리기, 시스템들간의 DTD 정보 교환을 위한 정보 테이블 관리기, XOMT도에 의해 생성된 DTD 객체들의 위치를 관리하는 다이아그램 위치 관리기로 구성된다.

정보 테이블 관리기는 사용자 화일 관리기에 의해 생성, 수정된 DTD 정보를 임시로 저장하고 다이아그램 위치 관리기에서의 위치 조작을 위한 정보 제공자 역할을 수행한다. 또한 텍스트 DTD의 XOMT도 변환을 위한 임시 정보 저장소 역할을 수행한다.

다이아그램 위치 관리기는 XOMT도 위치 정보를 해석하여 DTD로 쉽게 변환하기 위한 중간 단계의 DTD 정보를 관리하는 XOMT 테이블을 생성한다. 이는 위치 정보 및 순수 XOMT도 정보를 저장하는 위치 분리기, 사용자의 요구시 저장된 위치 정보를 결합시키는 위치 결합기, 텍스트 DTD에 대한 XOMT도의 위치를 자동 생성하는 위치 정보 자동 생성기로 구성된다.

<그림 4>는 XDMS를 OMT의 객체도로 표현한 것이다.



<그림 4> XDMS의 객체도

4.2 XOMT도-DTD 변환 부시스템(XDTS)

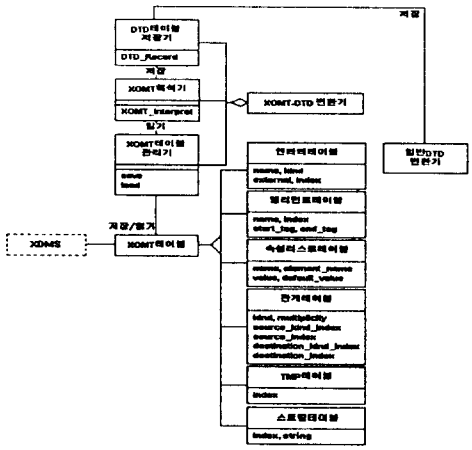
XOMT도-DTD 변환 부시스템(XOMT-DTD Transformation Subsystem, XDTS)은 XOMT도로 설계되어 XDMS에 의해 객체 지향 데이타베이스에 저장된 SGML DTD 정보를 읽어내어 텍스트 형태의 DTD로 변환한다. 이를 위해 XDTS는 XOMT 테이블 관리기, XOMT도 해석기, DTD 테이블 저장기로 구성된다.

XOMT 테이블 관리기는 해당 DTD로 변환될 구성요소인 엔티티, 엘리먼트, 속성리스트, 관계, TMP, 엔티티, 문자열 등의 정보가 저장되어 있는 정보 테이블을 관리하는 역할을 담당한다.

XOMT도 해석기는 XOMT도 정보 테이블로부터 저장된 정보를 읽어 완전한 텍스트 DTD로 변환하기 위해 관련된 관계 정보를 탐색하여 해석한다.

DTD 테이블 저장기는 객체 지향 데이타베이스에서 DTD 정보를 관리하기 위한 메타 스키마로 사상하기 전 변환된 DTD를 레코드 단위로 한 건씩 저장하는 부분이다.

<그림 5>는 XDTS를 OMT의 객체도로 표현한 것이다.



<그림 5> XDTS의 객체도

사용자에 의해 설계된 XOMT도를 해당 SGML DTD로 변환하는 과정을 개괄적으로 기술하면 다음과 같다.

[단계1] XOMT 테이블에 저장되어 있는 각 XOMT도 정보 선택

변환되는 정보들은 위치 정보가 분리된 객체 정보로써 엔티티, 엘리먼트, 속성리스트 테이블에 각각 저장되어 있다가 변환을 위해 순차적으로 읽혀진다. 다음은 엔티티 XOMT 테이블로부터 읽혀지는 엔티티 관련 정보를 나타내기 위해 클래스를 정의하고 앞에 보인 <그림 1>의 XOMT도에서 실제 발생된 객체 $_i$의 예를 보인 것이다.

EntityXOMTClass(Kind:{0,1), Name:String, External:{0,1,2}, Index:integer);
EntityXOMTClass$_i$(Kind:0, Name:'details', External:0, Index:1);

[단계2] 내용 모델을 구성하기 위해 선택한 각 정보와 관련된 관계 정보를 관계 테이블에서 탐색하여 변환

변환하기 위해 읽은 객체 정보들 중 엔티티/엘리먼트는 해당 인덱스와 일치하는 source_index를 가진 관계 정보를 관계 테이블에서 탐색한다. 이 때 속성리스트는 선택하여 읽혀진 객체 정보들을 그대로 해당 DTD 테이블에 저장한다. 또한 두 개 이상의 하위 엘리먼트들이 묶여져서 발생 지시자와 함께 나타나는 것을 표현하기 위해 사용한 임시 엘리먼트 클래스인 TMP 정보가 탐색되었을 때는 관계 테이블을 재참조하여 관련된 관계 정보를 계속해서 탐색한다. 다음은 위에서 읽혀진 엔티티 객체 $_i$의 관계 정보를 탐색한 결과 $_l$, $_m$의 예를 나타낸 것이다.

RelationXOMTClass(Kind:{0~6), Sequence:Integer, Source_Kind_index:Integer, Source_index:Integer, Destination_Kind_index:Integer, Destination_index:Integer, Multiplicity:{0,1,2,3,));
RelationXOMTClass$_l$(Kind:1, Sequence:1, Source_Kind_index:3, Source_index:1,Destination_Kind_index:1,Destination_index:1, Multiplicity:0);
RelationXOMTClass$_m$(Kind:1, Sequence:1, Source_Kind_index:3, Source_index:1,Destination_Kind_index:1,Destination_index:2, Multiplicity:0);

[단계3] 해석·연결 알고리즘에 따라 완전한 텍스트 DTD 완성

탐색이 완료되면 한 건의 엔티티나 엘리먼트에 대해 탐색된 관계 정보들을 연결하는 연결자('|', '&', ',')와 발생 지시자('*', '+', '?')를 해석하여 내용 모델로 구성함으로써 완전한 하나의 텍스트 DTD를 완성한다.

[단계4] 변환된 DTD를 데이타베이스에 저장

이렇게 하여 변환된 DTD는 이들을 관리하는 DTD 데이타베이스에 사상하기 위해 엔티티, 엘리먼트, 속성리스트 테이블에 저장한다. 다음은 앞서 보인 <그림 1>의 XOMT도를 변환하여 각 DTD 테이블에 저장한 결과이다.

ENTITY			
KIND	NAME	EXTERNAL	CONTENT
'%'	'details'	' '	'Q|Pref'

ELEMENT			
EL_NAME	S_TAG	E_TAG	CONTENT
'Pref'	'-'	'O'	'EMPTY
'Q'	'-'	'-'	'#PCDATA
'P'	'-'	'O'	'(#PCDATA|%details;)*'
'Close'	'-'	'O'	'#PCDATA'
'Body'	'-'	'O'	'P*'
'Memo'	'-'	'-'	'((To&From),Body,Close?)'
'(To|From)'	'-'	'O'	'#PCDATA'

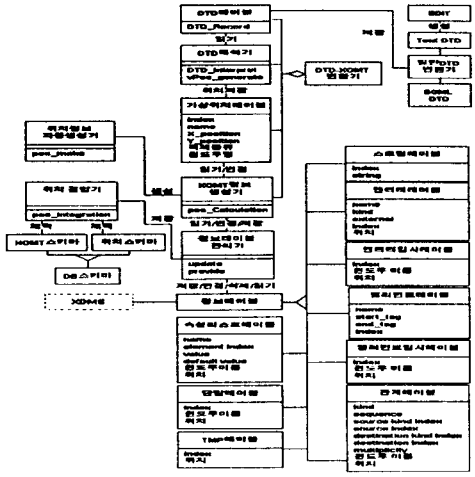
ATTLIST			
EL_NAME	NAME	VALUE	DEFAULT_VALUE
'Pref'	'refid'	'IDREF'	'#REQUIRED'
'P'	'id'	'ID'	'#IMPLIED'
'Memo'	'STATUS'	'confiden|public'	'public'

4.3 DTD-XOMT도 변환 부시스템(DXTS)

DTD-XOMT도 변환 부시스템(DTD-XOMT Transformation Subsystem, DXTS)은 XDMS의 위치 분리기에 의해 순수 DTD 객체 정보와 위치 정보로 분리되어 저장된 XOMT도 정보를 위치 결합기의 위치 결합 기능을 이용하여 XOMT도로 재변환한다.

이때, XOMT 객체 정보는 OID로서 위치 정보를 상속받기 때문에 데이타베이스의 조작만으로도 위치 결합이 가능하다. 그러나 텍스트 형태로 입력된 DTD의 XOMT도 변환은 위치 정보가 존재하지 않기 때문에 보다 복잡한 알고리즘이 필요하다. 이를 위해 DXTS는 레코드 단위의 DTD 정보를 저장하는 DTD 테이블, DTD의 해석을 위한 DTD 해석기, 해석된 객체들의 가상 위치를 저장하는 가상 위치 테이블, XOMT도로 변환할 수 있게 객체의 위치 생성을 위한 제어와 정보 테이블에 저장된 정보의 관리를 위한 XOMT 정보 생성기로 구성된다.

<그림 6>은 DXTS를 OMT의 객체도로 표현한 것이다.



<그림 6> DXTS의 객체도

다음은 DXTS에서 실제 텍스트 형태로 입력된 SGML DTD를 XOMT도로 변환하는 과정이다.

[단계1] DTD의 분리

SGML 문서 편집기에 의해 생성된 DTD를 일반 DTD 변환기로 해석하여 엔티티, 엘리먼트, 속성리스트로 분리하여 DTD 테이블에 저장한다.

[단계2] DTD의 해석 및 우선 순위 부여

DTD 해석기는 DTD 테이블에 저장되어 있는 엘리먼트 레코드부터 해석을 시작하며 이 때 해석된 정보들은 정보 테이블 관리기에 의해 정보 테이블에 저장된다. 그리고 해석 순서와 구분자(delimiter)에 의해 우선 순위가 결정되며 이 우선 순위는 가상 테이블에 객체의 종류, 인덱스, 그리고 윈도우명과 함께 저장된다. 그러나 XOMT도의 표기상 TMP 객체의 처리가 수반되므로 해석 중간 과정에서 이를 수행한다.

[단계3] 실제 위치 정보 생성 및 결합

DTD 테이블의 해석이 끝나면 XOMT 정보 생성기는 가상 위치 테이블의 정보와 각 객체들의 관계 정보를 저장한 관계 테이블의 내용을 비교 분석하면서 위치 정보 자동 생성기가 가지고 있는 각 객체의 크기 정보를 결합시켜 윈도우상의 실제 위치를 생성시켜 해당 정보 테이블에 저장한다. 이때 윈도우상의 실제 위치 생성시 XOMT도로 표현된 전체 그림

의 안정도를 위해 각 객체들의 위치와 크기 정보를 적절히 조절하여 변경해야 하며 이를 위해 몇 가지 알고리즘이 요구된다.

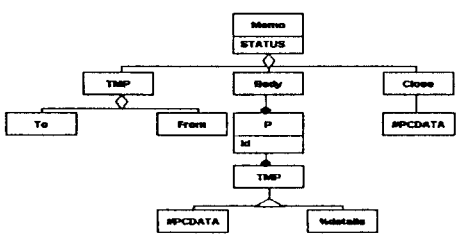
(1) 전체 윈도우에서 상위의 첫 번째 그려지는 엘리먼트를 탐색하여 시작 위치를 부여한다.
(2) 엘리먼트가 속성리스트를 포함할 경우는 해당 엘리먼트의 위치(X좌표)와 같게 부여한다.
(3) 내용 모델로 구성되는 하위 엘리먼트들의 각 위치는 탐색된 하위 엘리먼트의 수에 따라 계산하여 부여한다.
(4) 기존 엘리먼트가 다른 여러 엘리먼트의 내용 모델로 구성되어 재참조될 경우는 그려질 실제 위치를 감안하여 위치(Y좌표)를 내려준다.
(5) 혼합된 내용 모델을 표현하는 TMP는 TMP 객체가 그려질 위치를 먼저 부여한다.

다음은 실제 위치 정보를 자동 생성하여 XOMT도를 그리기 위해 만들어낸 그림 정보에 대한 사용자 화일 중 엘리먼트 테이블을 나타낸 것이다.

작업 엘리먼트 화일					
Index	Window_Name	X	Y	Length	Width
1	details	36	-86	60	25
2	details	125	-86	60	25
3	Main	235	-30	60	25
4	Main	235	-117	60	25
5	Main	396	-117	60	25
6	TMP1	30	-42	60	25
7	TMP2	30	-98	60	25
6	Main	35	-190	60	25
7	Main	121	-190	60	25
8	Main	235	-190	60	25

[단계4] XOMT도 생성

실제 위치 정보 생성 후 정보 테이블에 저장된 XOMT도의 정보를 사용자 화일 관리기로 전달함으로써 XOMT도가 생성된다. <그림 7>은 앞서 보인 텍스트 형태의 메모 DTD에 대해 자동 생성된 위치 정보와 결합하여 XOMT도로 변환한 결과이다.



<그림 7> 위치 생성 후 그려진 XOMT도

5. 결론 및 향후 연구 과제

본 논문은 XOMT도를 이용하여 설계된 SGML DTD를 객체 지향 데이타베이스를 통해 공유하고 관리할 수 있도록 이들 사이에 상호 변환하는 시스템인 TSXD를 설계하고 구현한 내용을 기술하였다. TSXD는 사용자가 전문적인 지식없이도 쉽게 DTD를 설계하고, 이들을 통합적으로 데이타베이스에서 관리할 수 있게 한다. TSXD의 구현 결과 동일한 SGML DTD의 경우라도 표현의 정확도면에서 다소 차이가 있음을 알 수 있었다. 이는 추후 보다 정확한 위치 생성 방법에 대한 연구를 보강함으로써 해결할 수 있을 것이다.

앞으로 시간성을 표현할 수 있는 HyTime[6] 문서도 관리할 수 있도록 XOMT를 확장하고 이를 기반으로 HyTime DTD 설계도와 그것의 데이타베이스 사이에 변환하는 시스템의 연구도 요구된다.

참고문헌

[1] O. Deux et al., "The O2 System," Communications of the ACM, Vol. 34, No. 10, pp. 34-48, October 1991.

[2] W. Kim, "Object-Oriented Database : Definition and Research Directions," IEEE Transactions on Knowledge and Data Engineering, Vol. 3, No. 3, pp. 327-431, Sept. 1990.

[3] 박인호, 한에노, 강현석, 배종민, 정은주, 김은정, 김완석, "XOMT : SGML DTD 설계를 위한 객체 다이아그래밍 기법," 한국정보과학회 논문지, 제3권, 제3호, pp. 228-237, 1997. 6.

[4] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, Object-Oriented Modeling and Design, Prentice-Hall, 1991.

[5] E. Herwijen, Practical SGML, 2nd Edition, Kluwer Academic Publishers, 1994.

[6] ISO, Hypermedia/Time-based Structure Language : HyTime(ISO 10744), 1992.