객체 지향 데이타베이스를 이용한 HyTime 문서의 구조 정보 관리

박인호[†] · 강현석[††]

요 약

하이퍼미디어 응용에 나타나는 다양한 형태의 멀티미디어 데이타와 이들 사이의 동기화 등을 지원하기 위해 하이퍼미디어 전자 문서의 기술 표준인 HyTime(Hypermedia/Time-based Structuring Language)이 사용되고 있다. 그런데 이러한 HyTime 문서를 효과적으로 관리하기 위해서는 HyTime 문서의 논리적 구조 정보를 체계적으로 데이타베이스에 저장하여 여러 사용자가 공유할 수 있도록 관리할 필요가 있다.

본 논문에서는 하이퍼미디어 문서의 논리적인 구조를 정의하는 HyTime DTD(Document Type Definition)를 객체 지향 데이타베이스에서 체계적으로 관리하기 위해 메타 데이타베이스 스키마를 설계하고 이를 관리하는 방법을 기술한다.

Management of the Structure Information of HyTime Documents using Object-Oriented Database

In-Ho Park[†] and Hyun-Syug Kang[††]

ABSTRACT

HyTime(Hypermedia/Time-based Structuring Language), an international standard language to describe hypermedia electronic documents, is used to support the synchronization between various multimedia data for hypermedia applications. To manage the HyTime documents efficiently for shared environment, the logical structure information of them should be managed by database in a systematic way.

In this paper, we design a meta-database schema of HyTime DTDs(Document Type Definition) which define the logical structure of hypermedia documents and show how to manage the meta-database schema for storing the HyTime DTDs in the object-oriented database.

Key words: HyTime, OODB, 문서 관리, 구조 정보

1. 서 론

최근 멀티미디어와 인터넷 기술이 급속히 발전하면서 전자 도서관, 원격 교육, VOD 등과 같은 하이퍼미디어 응용 서비스가 활발하게 제공되고 있다. 하이퍼미디어 응용에서는 텍스트, 이미지, 오디오, 비디오 등과 같이 다양한 미디어들을 처리해야 하고 이들 사이에 동기화를 지원해야 한다. 이에 따라 하이퍼미디어(Hypermedia) 전자 문서를 기술하는 표준 언어로 HyTime(Hypermedia/Time based Structuring Language : ISO 10744)이 사용되고 있다[1,2]. HyTime으로 기술되는 하이퍼미디어 문서를 데이타베이스에서 효율적으로 관리하게 되면 급증하는 하이퍼미디어 응용을 보다 쉽고 정확하게 다룰 수 있다. 그런데 이러한 HyTime 전자 문서는 다양한 미디어들이 복잡한 구조로 결합되어 있을 뿐만 아니라 다양한

본 논문은 2000년도 정보통신부의 대학기초연구 지원사업의 결과입니다.

[†] 경상대학교 컴퓨터과학과

[††] 종신회원, 경상대학교 컴퓨터과학과/컴퓨터정보통신연구소 교수

사용자 자료형을 요구하기 때문에 기존의 관계형 데이타베이스 시스템들 보다는 객체 식별성, 상속성, 추상화, 다형성 등 객체 지향 개념들을 지원하는 객체 지향 데이타베이스 시스템에서 관리하는 것이 효과적이다[3,4].

한편, 이러한 하이퍼미디어 문서를 통합 관리하기 위해서는 문서의 논리 구조를 정의하는 HyTime DTD(Document Type Definition)를 설계할 수 있는 방법론과 이 DTD를 기준으로 HyTime 문서를 작성할 수 있는 편집기, 브라우져, 그리고 이들 DTD와 문서를 데이타베이스에 통합적으로 저장하여 검색하고 공유할 수 있는 기법이 필요하다.

따라서 우리는 HyTime을 이용한 하이퍼미디어 문서의 관리를 위해 HyTime DTD를 시각적으로 설계하기 위한 방법론에서부터 다양한 조건의 검색과 최소 단위 공유가 가능하도록 하이퍼미디어 문서를 저장하기 위한 데이타베이스 스키마의 설계까지를 포함한 하이퍼미디어 통합 관리 시스템을 구축하고자 한다. HyTime 문서를 데이타베이스에서 통합적으로 관리하기 위해서는 우선 하이퍼미디어 문서의 논리적인 구조가 정의되어 있는 HyTime DTD를 체계적으로 데이타베이스에 저장하여 여러 사용자들이 공유할 수 있도록 관리할 필요가 있다. 즉, HyTime 문서의 논리적인 구조가 정의된 HyTime DTD를 데이타베이스로 관리함으로써 문서 구조의 공유와 검색 방법을 제공할 수 있다. 또한 HyTime 문서 자체를 관리하는 문서 데이타베이스 스키마를 동적으로 생성할 수 있고 문서 전체를 저장하는 것이 아니라 엘리먼트 단위로 세분하여 관리함으로써 내용의 공유와 구조 검색 및 내용 검색을 위한 방법론을 제공할 수 있다. 이를 위해 본 논문에서는 HyTime 문서의 논리적인 구조를 효율적으로 관리할 수 있는 HyTime DTD 메타 데이타베이스 스키마를 설계하고 이의 관리 방법을 기술한다.

논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대하여 알아보고, 3장에서는 SGML과 HyTime에 대해 간단하게 요약한다. 4장에서는 하이퍼미디어 전자 문서의 논리적인 구조를 관리하는 HyTime DTD 메타 데이타베이스 스키마를 제안하고, 5장에서는 HyTime DTD를 메타 데이타베이스 스키마에 사상하는 방법을 기술한다. 마지막으로 6장에서 결론 및 향후 계획에 대해 논의한다.

2. 관련 연구

하이퍼미디어 문서 관리를 위한 연구는 비교적 활발히 진행되고 있다. Balasubramaniam[5]은 하이퍼미디어의 관리에 관계형 데이타베이스가 문제가 있음을 보이고 구조화된 멀티미디어 문서의 검색과 저장에 객체 지향 개념을 적용한 HyperBase라는 객체 지향 데이타 모델을 사용하였으나 관계형 데이타베이스인 SyBase에서 구현되었다. Böhm[6]은 HyTime을 다루기 위해 D-STREAT의 확장을 제시하였는데, DTD의 특징들을 두개의 클래스 계층으로 나누어서 다룬다. 여기에는 TERMINAL과 NONTERMINAL이라는 두개의 메타 클래스가 있는데 TERMINAL 클래스는 문서 계층에서 종말 노드에 관련되고 NONTERMINAL은 구조화된 엘리먼트에 관련된다. 이 방법의 단점은 엘리먼트가 의미를 가지지 않는다는 것이다. 김현기[7]는 Postgres 데이타베이스를 하부 저장 구조로 사용한 객체 지향 하이퍼미디어 시스템 및 SGML을 이용한 하이퍼미디어 기술 언어 HOML(Hypermedia Object Modeling Language)을 설계하여 정보 검색 및 데이타베이스 질의 기능을 제공한다. Özsu[8]는 HyTime DTD, 문서와 멀티미

표 1. 하이퍼미디어 데이타베이스 시스템들의 비교

시스템	특징	장단점
HyperBase[5]	객체 지향 데이타 모델 제시	RDBMS(SyBase)에서 구현
D-STREAT[6]	문서의 태그 정보와 내용을 구분하여 저장하기 위해 두 개의 클래스 이용	문서의 저장에는 용이하지만 구조 검색 및 내용 검색을 위해서는 불합리
HOML[7]	하이퍼미디어 기술 언어 HOML을 설계하고 정보 검색 및 데이타베이스 질의 기능 제공	정보 검색 및 질의 기능을 위해 문서 데이타베이스를 전부 검색하는 단점
Type System[8]	타입 시스템을 정의하여 멀티미디어 데이타와 DTD의 엘리먼트를 구분. 하이퍼미디어를 위한 저장 구조	하이퍼미디어 문서의 저장에는 용이하지만 구조 검색 및 내용 검색을 위해서는 불합리

디어 데이타들을 멀티미디어 데이타베이스에서 관리하기 위하여 타입 시스템(Type System)들을 제안하였다. 이 타입 시스템은 DTD와 문서 사이에 위치하며 멀티미디어 데이타들을 관리하는 Atomic 타입 시스템과 엘리먼트의 기능별로 구분한 엘리먼트 타입 시스템으로 크게 구분할 수 있다. 이 시스템은 DTD를 기준으로 타입 시스템을 생성하고 타입 시스템을 기준으로 문서와의 연결 구조를 정의하고 있다. 하이퍼미디어 데이타베이스 시스템들을 비교하면 표 1과 같다.

그런데 이러한 시스템들은 하이퍼미디어의 관리를 위해 특정 분야만을 지원하고 있다. 즉, 관계형 데이터 모델이나 객체 지향 데이터 모델을 이용하여 문서의 태그와 그 내용을 구분하여 단순히 저장하고 검색하는 방식만을 제공할 뿐 문서의 구조를 설계하는 것에서부터 응용까지의 하이퍼미디어 문서 전반에 걸친 관리는 지원하지 않는다. 본 논문에서는 OODBMS를 이용하여 HyTime DTD 관리를 위한 객체 지향 데이터 모델을 제안하고, DTD의 구조를 최소 단위로 분할하여 관리하고 또한 이들을 결합하여 캡슐화함으로써 DTD의 논리 구조를 유지하면서 관리할 수 있도록 한다. 이렇게 함으로써 DTD의 관리가 보다 지능적이고 다양한 방식의 질의(powerful query)가 가능하다. 또한, HyTime 문서 관리 데이터베이스 스키마의 동적인 생성을 지원하는 하이퍼미디어 문서 통합 관리 시스템을 구축하기 위해 HyTime 문서의 논리적인 구조를 효율적으로 관리할 수 있는 HyTime DTD 메타 데이터베이스 스키마를 설계하고 이를 통해 DTD 정보를 동적으로 관리하는 방법을 제시한다.

3. SGML과 HyTime

3.1 SGML

최근 각종 정보들이 전자 문서화되면서 이들을 이기종 시스템들 간에 처리하고 교환할 수 있게 하는 것이 중요하게 되었다. 이를 위해서 전자 문서는 그것의 내용 외에도 내용의 배치를 위한 정보가 필요한데 이런 추가적인 정보를 전자 문서에 삽입하는 것을 마크업(markup)이라 하며, 이를 기술하는 메타 언어로 SGML(Standard Generalized Markup Language)이 제안되었다[9]. SGML 전자 문서를 작성하기 위해서는 문서의 논리적 구조를 나타내는 DTD를 우선 기술하여야 한다.

3.2 HyTime

HyTime[1,2]은 하이퍼미디어 문서의 효율적인 관리를 위해 SGML로 정의된 하나의 응용 구조이며, SGML의 구문을 사용하여 기술된 메타 DTD로서 HyTime에 적합한 모든 응용은 SGML에 적합한 응용이 된다. 또한, 아키텍쳐 폼(architectural forms : AF)이라고 불리어지는 틀의 집합(set of templates)인 메타 문서 형식(meta document type)이다. HyTime 문서의 기본 구조를 기술하는 메타 DTD의 규칙 집합은 아키텍쳐 폼에 의해 표현된다. HyTime은 다양한 응용들의 요구를 충족시키기 위해 체계적인 모듈 분리를 하고 있다. HyTime 아키텍쳐 폼은 6개의 모듈(Base, Measurement, Location Address, Hyperlinks, Scheduling, Rendition module)로 구성되며, 각 모듈은 기본적인 기능 집합과 하나 이상의 선택적인 특징을 포함한다.

3.3 HyTime의 선언

HyTime을 지원하기 위한 SGML의 확장은 문서 내에 HyTime의 아키텍쳐 폼들을 연결해야 한다. 이

```
FORMAL YES
APPINFO ArcBase
<!AFDR "ISO/IEC 10744:1992" >
<?ArcBase HyTime >
<!Notation HyTime PUBLIC "ISO/IEC 10744:1992//NOTATION HYTIME ARCBASE
                    HyTime Architecture Definition Document//EN" >
<!Attlist #NOTATION HyTime
          ArcFormA   NAME  HyTime
          ArcNamrA   NAME  HyNames
          ArcSuprA   NAME  HyBrid
          ArcDocF    NAME  HyDoc
          ArcDTD     CDATA "HyTime"
          ArcBridF   NAME  HyBrid
          ArcAuto    (ArcAuto|nArcAuto)  ArcAuto
          ArcIndr    (ArcIndr|nArcIndr)  nArcIndr
          ArcOptSA   NAMES "base locs links"
          base       CDATA "base autobos exidrefs"
          locs       CDATA "locs anydtd anysgml mixcase
                      multiloc bibloc nameloc"
          links      CDATA "links" >
<!NOTATION AFDRMeta PUBLIC "ISO/IEC 10744//...//EN" >
<!Entity HyTime PUBLIC "ISO/IEC 10744//...//EN"  CDATA AFDRMeta >
.................
```

그림 1. BibCat 메타 DTD의 일부

메카니즘은 노테이션이나 엔티티를 이용하여 문서와 아키텍쳐 폼을 연결하고 문서에 사용되는 모듈의 집합이나 아키텍쳐 폼을 선언한다[1,2]. 구체적으로 기술하면, SGML 선언부의 FORMAL 파라메타 YES는 외부 객체와의 연결을 의미하며, APPINFO 파라메타에 "APPINFO ArcBase" 혹은 "APPINFO HyTime"을 선언하는 것은 HyTime 아키텍쳐 폼을 문서에 사용하는 것을 의미한다. 그림 1은 노테이션과 엔티티를 이용하여 HyTime 아키텍쳐 폼과 연결한 예로써 노테이션의 속성부에서 DTD 내에 사용될 명세를 기술하고 있다. <!AFDR "ISO/IEC 10744: 1992"> 이라고 선언하면 AFDR(Architectural Forms Definition Requirements)의 사용을 기술한 것으로서, 이것은 HyTime을 이용한 새로운 메타 DTD를 생성한다는 의미이다.

4. HyTime DTD 메타 데이타베이스 스키마

HyTime은 SGML 문법을 따르기 때문에 문서의 구조를 정의하는 DTD의 기술이 반드시 필요하며 DTD의 구조를 유지하면서 공유할 수 있도록 데이타베이스에서 관리할 필요가 있다. HyTime은 기본적으로 SGML의 응용이며 확장으로 SGML 메타 데이타베이스 스키마[10]를 확장하여 사용할 수 있다. 이 장에서는 SGML 메타 데이타베이스 스키마를 HyTime DTD도 관리할 수 있도록 선언부와 추가적인 부분을 확장하고 모든 HyTime DTD의 구문을 표현할 수 있도록 확장하였으며, 하이퍼미디어의 관리를 위한 메타 정보를 정의하고 있는 HyTime 아키텍쳐 폼들의 요소들을 관리하기 위해 HyTime 지원 데이타베이스 스키마를 설계한다. HyTime DTD의 구문은 SGML을 따르고, 하이퍼미디어 제어 정보는 HyTime 지원 데이터베이스 스키마에서 관리하기 때문에 모든 HyTime DTD의 관리가 가능하고 요소(component)별 검색이 가능하게 된다.

4.1 HyTime을 위한 SGML DTD 메타 데이타베이스 스키마의 확장

SGML 메타 데이타베이스 스키마는 SGML의 DTD를 저장하기 위한 구조를 가지고 있다. 따라서 이를 이용해 HyTime DTD를 저장하기 위해서는 우선 HyTime 문서를 인식할 수 있는 선언부를 저장할

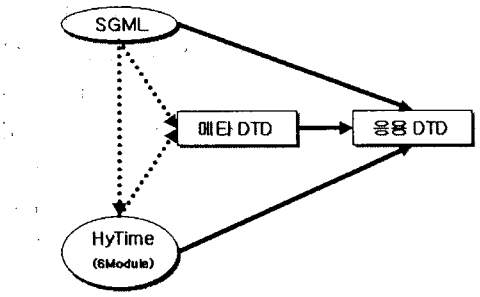

그림 2. HyTime DTD 계층도

수 있도록 확장되어야 한다. 메타 데이타베이스 스키마에 저장되는 HyTime DTD의 유형은 그림 2와 같이 크게 두 종류이다. 그림 2에서 점선으로 표시된 것과 같이 생성되는 유형이 있는데 이러한 DTD들은 메타 DTD로 다른 DTD를 생성하는데 기준이 되는 DTD이다. 또 실선 부분과 같이 메타 DTD나 직접 언어로부터 실제 문서를 작성하는 응용 DTD가 생성되는 유형이다.

그림 3은 SGML DTD를 데이타베이스에서 관리

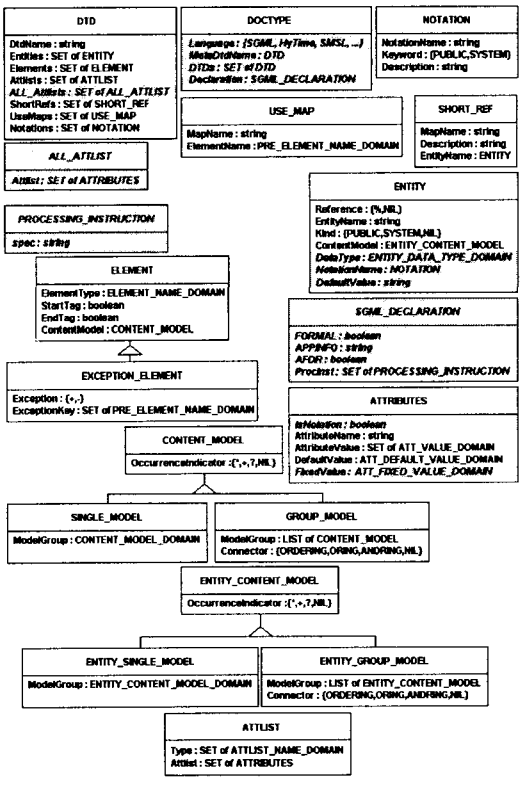

그림 3. HyTime DTD 메타 데이타베이스 스키마 객체도

하기 위해 설계된 SGML 메타 데이타베이스 스키마를 HyTime DTD도 관리할 수 있도록 확장하여 설계한 메타 데이타베이스 스키마로서 OMT 객체도[11]를 이용하여 표현한 것이다. 우리는 이러한 HyTime DTD를 데이터베이스에서 관리할 때 구성 요소별로 분리한 다음 그들 사이의 관계를 유지하는 방법을 사용한다. 즉, DTD를 단순한 문자열로서 관리하는 것이 아니라 DTD 문법에 따라 토큰화하여 구성 요소 사이의 종속 관계를 클래스 객체들간의 링크로 표현함으로써 DTD 계층 구조와 중첩 구조를 유지할 수 있을 뿐만 아니라 DTD를 기준으로 작성된 실제 문서의 내용을 관리하는 문서 관리 데이타베이스 스키마의 자동 생성이 가능하게 한다.

HyTime DTD의 구성 요소들을 각각 7개의 클래스들(DOCTYPE, ENTITY, ELEMENT, NOTATION, ATTLIST, SHORTREF, USEMAP)로 정의하고 HyTime 문서 선언을 위한 SGML 선언부 클래스(SGML_DECLARATION)를 기본 클래스로 정의하였다. 그리고 기술된 구성 요소들의 구문에 해당하는 파라미터들은 각 클래스의 속성들로 표현하였다.

SGML 메타 데이타베이스 스키마와 비교하여 그림 4는 클래스 ENTITY와 클래스 ATTRIBUTES를 HyTime 구문에 부합되게 확장하였으며, SGML을 기반으로 하는 마크업 언어들을 포괄적으로 수용하기 위해 DOCTYPE 클래스를 확장하고 클래스 SGML_DECLARATION을 새롭게 추가하였다. 클래스 DOCTYPE에서는 DTD 설계를 위해 사용된 언어를 구분하기 위한 속성으로 Language를 정의하였는데 이는 SGML을 이용한 여러 마크업 언어들을 위해 확장이 가능하다. 그리고 그림 2의 HyTime DTD 계층도에 나타난 것처럼 사용자가 설계한 메타 DTD명이나 언어로부터 직접 설계한 응용 DTD명을 나타내는 속성으로 MetaDtdName, 메타 DTD를 이용한 응용 DTD의 설계일 경우 DTD 집합을 나타내는 속성으로 DTDs, DTD의 선언부를 나타내는 속성으로 Declaration을 가진다. 또한 메타 DTD에 포함된 모든 엘리먼트에 일괄적인 속성 지정을 위해 ALL_ATTLIST 클래스를 정의한다. 이러한 속성리스트는 해당 엘리먼트 이름부에 '#ALL'이라는 키워드를 이용하여 정의한다. HyTime DTD의 논리 구조를 유지하면서 데이타베이스에서 관리하기 때문에 HyTime 문서를 위한 문서 데이타베이스 스키마를 동적으로 생성할 수 있고 다양한 조건 검색이 가능하다는 장점이 있다. SGML DTD를 위한 메타 데이타베이스 스키마에서 확장된 부분은 그림 3에서 이탤릭체와 음영으로 표시하였다.

그림 3의 메타 데이타베이스 스키마에서는 클래스의 속성 도메인에 나타날 수 있는 타입들이 무척 다양하고 복잡하다. 따라서 이를 효과적으로 다루기 위해 DTD의 구성 요소로서 나타나는 클래스 계층 구조와는 별도로 속성 도메인 타입 계층 구조를 설계한 것이 그림 4인데, 추가된 부분은 그림 4에서 이탤릭체와 음영으로 표시하였다.

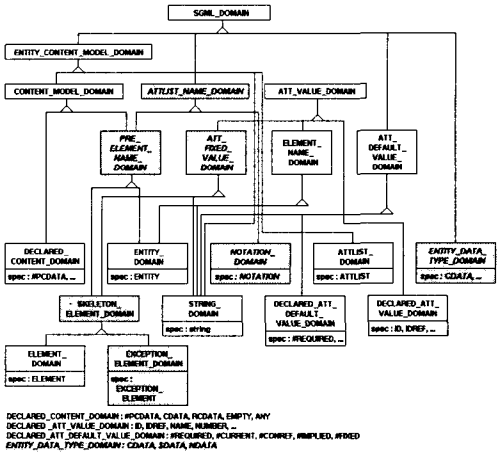

그림 4. HyTime DTD 속성 도메인 클래스 타입 계층도

4.2 HyTime 지원 데이타베이스 스키마 설계

HyTime 지원 데이타베이스 스키마는 HyTime에서 정의되어 있는 6개의 모듈을 데이타베이스에 관리하기 위한 스키마 구조이다. HyTime의 6개 모듈에서 정의되어 있는 아키텍쳐 폼은 상속으로 응용 DTD에서 사용할 수 있기 때문에 6개 모듈의 관리가 별도로 필요하다.

그림 5는 HyTime의 6개 모듈에 대한 정보를 관리하기 위한 데이타베이스 스키마이다. 기본 클래스 HYTIME_SECTION은 HyTime 구성 요소들(meta entity, meta element, meta attlist)을 하위 클래스로 가지는 HYTIME_ARCHITECTURAL_FORMS 클래스를 도메인으로 하는 속성을 가진다. HyTime 구성 요소들은 메타 데이타베이스 스키마의 인스턴스로 저장되고 HYTIME_SECTION 클래스를 상위 클래스로 하여 6개의 모듈들을 클래스로 정의하고

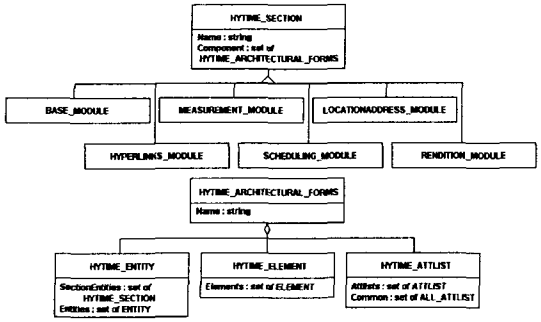

그림 5. HyTime 지원 데이타베이스 스키마

상위 클래스로부터 속성들을 상속받는다. HyTime 지원 데이타베이스 스키마는 DTD를 설계할 때 참조할 수 있도록 하고, HyTime 문서 자체를 데이타베이스에서 관리하기 위한 문서 관리 데이타베이스 스키마를 자동으로 생성할 때에도 참조할 수 있다.

5. HyTime DTD의 관리 방법

이 장에서는 예제 HyTime DTD를 4장의 메타 데이타베이스 스키마에 인스턴스로 사상시키는 과정을 알아봄으로써 HyTime DTD를 관리하는 방법을 기술한다. 본 논문에서 제안하는 데이터베이스 스키마의 구현 환경은 Unix Workstation에 ETRI에서 개발된 OODBMS인 바다Ⅲ로 구현하였으며, C++ 언어를 이용하여 설계하였다.

그림 6은 HyTime DTD를 HyTime 메타 데이터베이스 스키마에 사상하는 과정을 나타낸 개괄 구조도이다.

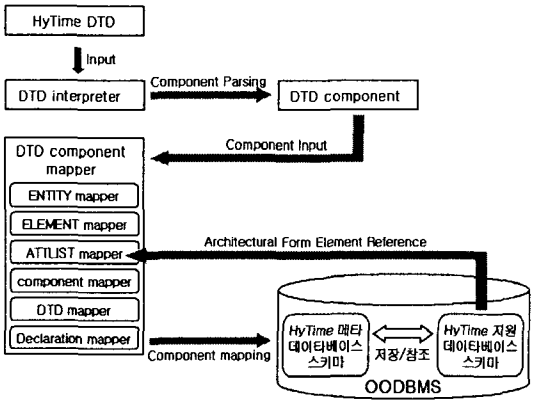

그림 6. HyTime DTD 사상기의 개괄 구조도

그림 6에서 HyTime DTD 요소들이 분해된 후, 각 요소별로 HyTime 메타 데이터베이스 스키마에 사상하는 DTD component mapper의 과정은 다음과 같이 7단계로 이루어진다.

〔단계 1〕 HyTime DTD의 엔티티들을 메타 데이터베이스 스키마에 사상

DTD에 나타난 엔티티를 클래스 ENTITY의 인스턴스로 사상시킨다. ENTITY 클래스의 정의는 아래와 같다.

Reference	(%, NIL)
EntityName	string
Kind	(PUBLIC, SYSTEM, NIL)
ContentModel	*ENTITY_CONTENT_MODEL*
DataType	*ENTITY_DATA_TYPE_DOMAIN*
NotationName	*NOTATION*
DefaultValue	string

엔티티의 대치 영역이 여러 개의 구성 요소들로 이루어져 있으면 그들을 하나로 묶어 클래스 ENTITY_GROUP_MODEL의 인스턴스로 발생시키고, 한 개의 구성 요소로 이루어져 있으면 클래스 ENTITY_SINGLE_MODEL의 인스턴스로 생성시킨 후 속성 ContentModel의 속성값으로 그에 해당되는 객체식별자를 기술한다. 속성 ContentModel의 도메인은 클래스 ENTITY_CONTENT_MODEL인데, 그림 3에서 클래스 ENTITY_CONTENT_MODEL은 클래스 ENTITY_GROUP_MODEL과 클래스 ENTITY_SINGLE_MODEL을 child로 가지는 parent 클래스로 정의되어 있다. 따라서 클래스 ENTITY_CONTENT_MODEL은 abstract 클래스가 된다.

〔단계 2〕 HyTime DTD의 엘리먼트들을 메타 데이터베이스 스키마에 사상

엘리먼트의 경우 예외 ELEMENT를 가지면 클래스 EXCEPTION_ELEMENT에, 그렇지 않으면 클래스 ELEMENT의 인스턴스로 사상된다. ELEMENT 클래스의 정의는 아래와 같다.

ElementType	*ELEMENT_TYPE_DOMAIN*
StartTag	boolean
EndTag	boolean
ContentModel	*CONTENT_MODEL*

내용 모델의 값이 한 개의 구성요소로 이루어져 있으면 클래스 SINGLE_MODEL에, 여러 개로 이루어져 있으면 클래스 GROUP_MODEL에 인스턴스를 발생시키고 그에 대한 객체식별자가 속성 ContentModel의 속성값이 된다. 클래스 CONTENT_MODEL은 클래스 SINGLE_MODEL과 GROUP_MODEL의 parent 클래스이다.

〔단계 3〕 정의되지 않은 엘리먼트, 엔티티, 예외 정보를 찾아 사상

DTD의 구성 요소 중 엔티티를 먼저 스키마에 사상하면서 엔티티의 대체 영역에 나타나는 엘리먼트와 엔티티의 객체식별자를 인스턴스에 사상시키지 못하는 문제가 발생한다. 따라서 단계 1과 단계 2에서 아직 발생하지 못한 객체(ELEMENT, ENTITY)를 클래스 UNDEFINED_ELEMENT와 클래스 UNDEFINED_ENTITY에서 임시로 보관하고 있다가 객체의 이름을 비교하여 실제 발생된 객체의 객체식별자를 얻은 후, 이를 해당 인스턴스의 속성값으로 사상한다.

〔단계 4〕 속성리스트를 메타 데이타베이스 스키마에 사상

속성리스트는 클래스 ATTLIST의 인스턴스로 발생된다.

〔단계 5〕 NOTATION, SHORTREF, USEMAP 등을메타 데이타베이스 스키마에 사상

NOTATION, SHORTREF, USEMAP 등을 메타 데이타베이스 스키마의 해당 클래스에 사상시킨다.

〔단계 6〕 발생된 클래스의 인스턴스들을 DTD 클래스에 사상

DTD의 구성 요소들을 스키마의 해당 클래스에 사상한 후, 발생된 인스턴스의 객체식별자를 DTD 구성 요소에 해당하는 속성에 속성값으로 사상시킨다. DTD 클래스의 정의는 위와 같다.

DtdName	string
Entities	*SET of ENTITY*
Elements	*SET of ELEMENT*
Attlists	*SET of ATTLIST*
ALL_Attlists	*SET of ALL_ATTLIST*
ShortRefs	*SET of SHORT_REF*
UseMaps	*SET of USE_MAP*
Notations	*SET of NOTATION*

〔단계 7〕 선언부의 내용을 클래스 SGML_DECLARATION에 사상한 후, 클래스 SGML_DECLARATION와 클래스 DTD의 인스턴스를 클래스 DOCTYPE에 사상

먼저 처리 명령을 사상한 후, 선언부의 내용을 클래스 SGML_DECLARATION의 인스턴스로 생성하고 클래스 DTD의 인스턴스와 함께 클래스 DOCTYPE에 사상한다.

그림 7은 예제 HyTime DTD인 BibCat 메타 DTD

```
FORMAL YES
APPINFO ArcBase
<!-- Meta-DTD for the BibCat architecture. -->
<!AFDR "ISO/IEC 10744:1992" >
<?ArcBase HyTime >
<!Notation HyTime PUBLIC "ISO/IEC 10744:1992//NOTATION HYTIME ARCBASE
                    HyTime Architecture Definition Document//EN" >
<!Attlist #NOTATION HyTime
         ArcFormA   NAME  HyTime
         ArcNamrA   NAME  HyNames
         ArcSuprA   NAME  HyBrid
         ArcDocF    NAME  HyDoc
         ArcDTD     CDATA "HyTime"
         ArcBridF   NAME  HyBrid
         ArcAuto    (ArcAuto|nArcAuto)  ArcAuto
         ArcIndr    (ArcIndr|nArcIndr)  nArcIndr
         ArcOptSA   NAMES "base locs links"
         base       CDATA "base autobos exidrefs"
         locs       CDATA "locs anydtd anysgml mixcase multiloc
                     bibloc nameloc"
         links      CDATA "links" >
<!NOTATION AFDRMeta PUBLIC "ISO/IEC 10744//...//EN" >
<!Entity HyTime PUBLIC "ISO/IEC 10744//,,,//EN" CDATA AFDRMeta >
<!ENTITY % fields "Title|Author|Publisher" >
<!ENTITY % leaf-content "#PCDATA|BibBrid|See-Also" >
<!ELEMENT BibDoc - O (GenTitle?, BibEntry*) +(Nameloc) >
<!ATTLIST BibDoc HyTime NAME #FIXED "HyDoc" >
<!ELEMENT BibBrid - - (%leaf-content;)* >
<!ELEMENT GenTitle - O (%leaf-content;)* >
<!ELEMENT BibEntry - O (%fields;)* >
<!ATTLIST BibEntry pubtype (book|serial|paper|film|article|undefined)
          undefined >
<!ELEMENT (%fields;) - - (%leaf-content;)* >
<!ATTLIST (%fields;) fieldno NUMBER #REQUIRED >
<!ELEMENT See-Also - O (%leaf-content;)* -(See-Also) >
<!ATTLIST See-Also HyTime   NAME  #FIXED "clink"
                   anchaddr IDREF #REQUIRED >
<!ELEMENT Nameloc - - (#PCDATA) >
<!ATTLIST Nameloc ID       ID     #REQUIRED
                  locsrc   ENTITY #IMPLIED
                  deflocsc (elements|entities) elements
                  HyTime   NAME   #FIXED "nameloc" >
<!-- End of BibCat meta-DTD -->
```

그림 7. HyTime DTD의 예(BibCat 메타 DTD)

```
notation1 : ('HyTime', PUBLIC, 'ISO/IEC 10744:1992//NOTATION HYTIME ARCBASE
                                HyTime Architecture Definition Document//EN');
attlist1 : (notation10id, [attributes10id, attributes20id, attributes30id, attributes40id, attributes50id,
  attributes60id, attributes70id, attributes80id, attributes90id, attributes100id, attributes110id, attributes120id]);
attributes1 : (T, 'ArcFormA', NAME, element10000id, NIL);
attributes2 : (T, 'ArcNamrA', NAME, element10010id, NIL);
attributes3 : (T, 'ArcSuprA', NAME, element10020id, NIL);
attributes4 : (T, 'ArcDocF', NAME, element10030id, NIL);
attributes5 : (T, 'ArcDTD', CDATA, 'HyTime', NIL);
attributes6 : (T, 'ArcBridF', NAME, element10040id, NIL);
attributes7 : (T, 'ArcAuto', ['ArcAuto', 'nArcAuto'], 'ArcAuto', NIL);
attributes8 : (T, 'ArcIndr', ['ArcIndr', 'nArcIndr'], 'nArcIndr', NIL);
attributes9 : (T, 'ArcOptSA', NAMES, 'base locs links', NIL);
attributes10 : (T, 'base', CDATA, 'base autobos exidrefs', NIL);
attributes11 : (T, 'locs', CDATA, 'locs anydtd anysgml mixcase multiloc bibloc nameloc', NIL);
attributes12 : (T, 'links', CDATA, 'links', NIL);
notation2 : ('AFDRMeta', PUBLIC, 'ISO/IEC 10744//NOTATION AFDR Meta-DTD Notation//EN');
entity1 : (NIL, 'HyTime', PUBLIC, entity_single_model10id, CDATA, notation20id, NIL);
entity_single_model1 : (NIL, 'ISO/IEC 10744//DTD AFDR Meta-DTD Notation//EN');
entity2 : (%, 'fields', NIL, entity_group_model10id, NIL, NIL, NIL);
entity_group_model1 : (NIL, [entity_single_model20id, entity_single_model30id, entity_single_model40id], ORING);
entity_single_model2 : (NIL, element50id);
entity_single_model3 : (NIL, element60id);
entity_single_model4 : (NIL, element70id);
entity3 : (%, 'leaf-content', NIL, entity_group_model20id, NIL, NIL, NIL);
entity_group_model2 : (NIL, [entity_single_model50id, entity_single_model60id, entity_single_model70id], ORING);
entity_single_model5 : (NIL, #PCDATA);
entity_single_model6 : (NIL, element20id);
entity_single_model7 : (NIL, exception_element20id);
exception_element1 : ('BibDoc', F, T, group_model10id, +, element80id);
group_model1 : (NIL, [single_model10id, single_model20id], ORDERING);
single_model1 : (?, element20id);
single_model2 : (*, element30id);
attlist2 : (exception_element10id, attributes130id);
attributes13 : (F, 'HyTime', NAME, #FIXED, element10030id);
element1 : ('BibBrid', F, F, single_model30id);
single_model3 : (*, entity30id);
element2 : ('GenTitle', F, T, single_model30id);
element3 : ('BibEntry', F, T, single_model40id);
single_model4 : (*, entity20id);
attlist3 : (element30id, attributes140id);
attributes14 : (F, 'pubtype', ['book', 'serial', 'paper', 'film', 'article', 'undefined'], 'undefined', NIL);
element4 : (entity20id, F, F, single_model30id);
element5 : ('Title', F, F, single_model30id);
element6 : ('Author', F, F, single_model30id);
element7 : ('Publisher', F, F, single_model30id);
attlist4 : (entity20id, attributes150id);
attributes15 : (F, 'fieldno', NUMBER, #REQUIRED, NIL);
exception_element2 : ('See-Also', F, T, single_model30id, -, exception_element20id);
attlist5 : (exception_element20id, [attributes160id, attributes170id]);
attributes16 : (F, 'HyTime', NAME, #FIXED, element10050id);
attributes17 : (F, 'anchaddr', IDREF, #REQUIRED, NIL);
element8 : ('Nameloc', F, F, single_model50id);
single_model5 : (NIL, #PCDATA);
attlist6 : (element80id, [attributes180id, attributes190id, attributes200id, attributes210id]);
attributes18 : (F, 'ID', ID, #REQUIRED, NIL);
attributes19 : (F, 'locsrc', ENTITY, #IMPLIED, NIL);
attributes20 : (F, 'deflocsc', ['elements', 'entities'], 'elements', NIL);
attributes21 : (F, 'HyTime', NAME, #FIXED, element10060id);
dtd1 : ('BibCat', [entity10id, entity20id, entity30id], [exception_element10id, element10id, element20id, element30id,
  element40id, element50id, element60id, element70id, exception_element20id, element80id], [attlist10id, attlist20id,
  attlist30id, attlist40id, attlist50id, attlist60id], NIL, NIL, NIL, [notation10id, notation20id]);
processing_instruction1 : ('ArcBase HyTime');
sgml_declaration1 : (T, 'ArcBase', T, processing_instruction10id);
doctype1 : (HyTime, dtd10id, NIL, sgml_declaration10id);
```

그림 8. BibCat 메타 DTD를 메타 데이타베이스 스키마에 사상한 결과

이고, 그림 8은 BibCat DTD를 HyTime 메타 데이타베이스 스키마에 사상한 후 발생된 인스턴스 객체들이다. 발생되는 각 인스턴스는 해당 클래스명을 소문자로 표시하고 발생하는 순서대로 첨자를 붙여서 나타내며, 객체식별자는 그 인스턴스에 첨자 'Oid'를 붙여서 나타낸다. 속성값이 여러 개의 집합일 경우는 대괄호([])를 사용하여 나타낸다. 그리고 6개 모듈에서 정의된 아키텍쳐 폼의 객체식별자들은 참조되는 순서대로 1000번부터 Oid를 부여하고 이탤릭체로 나타낸다.

6. 결론 및 향후 계획

본 논문은 HyTime 전자 문서를 효율적으로 관리하기 위한 객체 지향 데이타베이스 관리 방법을 제안하였다. 즉, HyTime DTD의 논리적인 구조를 관리하기 위해 메타 데이타베이스 스키마를 설계하고 이를 이용해 HyTime DTD 정보를 체계적으로 관리하는 방법을 보였다. 이렇게 HyTime DTD의 구조 정보를 유지, 관리함으로써 데이타베이스의 질의 처리, 재사용 등의 장점을 활용하고 자동적인 HyTime 문서 관리 데이타베이스 스키마의 생성이 가능하게 된다. 즉, 우리는 HyTime DTD 관리 기법을 SGML DTD 관리 기법의 확장으로 고안하였기 때문에 SGML 문서 관리 데이터베이스 스키마 생성 방법[10]을 그대로 사용할 수 있다. 그리고 우리는 이미 HyTime DTD 설계 방법론인 HOMT(HyTime support XOMT)를 제안한 바 있다[12]. HOMT는 HyTime DTD를 일반 사용자가 작성, 관리하기는 매우 어렵기 때문에 DTD의 구성요소를 다이아그램으로 정의하고 HyTime 모듈들을 참조할 수 있으며 트리 형식으로 DTD의 논리적인 구조를 관리할 수 있는 기법이다. 따라서 하이퍼미디어 응용 시스템 개발 시 객체 다이아그래밍 기법인 HOMT를 이용하여 SGML과 HyTime의 DTD를 쉽게 설계할 수 있으며, 설계된 DTD를 본 논문에서 제안한 방법을 이용하여 객체 지향 데이타베이스에서 통합적으로 관리할 수 있다.

이러한 통합 시스템을 통한 하이퍼미디어 문서의 관리는 현재 제기되고 있는 하이퍼미디어 문서 처리의 문제점인 가변적인 텍스트 엘리먼트의 다양한 크기, 서브 엘리먼트들의 공유, 분류 체계에 따른 관계 횡단에 대한 잦은 요구 등과 같은 어려움을 극복하는 해결책이 될 것이다.

현재 SGML과 HyTime을 기반으로 많은 종류의 언어들이 표준화되고 있다. 그 대표적인 것으로 DSSSL(Document Style Semantics and Specification Language, ISO/IEC 10179), SMSL(Standard Hypermedia/Multimedia Scripting Language, ISO/IEC 13240), SMDL(Standard Music Description Language, ISO/IEC 10743) 등이 있으며, 앞으로 이런 언어들로 기술된 DTD들도 모두 통합적으로 관리할 수 있는 시스템을 개발해야 할 것이다. 뿐만 아니라 실제 응용에 적용할 수 있는 엔진의 개발과 누구나 쉽게 접속하여 사용할 수 있도록 Web 기반 시스템으로의 확장도 추진되어야 할 것이다. SGML의 부분 집합(subset)으로 차세대 웹 문서의 표준인 XML(eXtensible Markup Language)[13]과의 연계도 고려되어야 한다.

참 고 문 헌

[1] ISO, *Hypermedia/Time-based Structuring Language : HyTime(ISO 10744)*, 1992.

[2] ISO/IEC, *Information Processing -- Hypermedia/ Time-based Structuring Language (HyTime) - 2nd edition*, 1997

[3] W. Kim, "Object-Oriented Database : Definition and Research Directions," IEEE Transactions on Knowledge and Data Engineering, Vol. 3, No. 1, pp. 327-341, Sept. 1990.

[4] J. Zhang, "Application of OODB and SGML Techniques in Text Database : An Electronic Dictionary System," SIGMOD RECORD, Vol. 24, No. 1, pp. 3-8, March 1995.

[5] V. Balasubramaniam, "State of the Art Review on Hypermedia Issues and Applications," Internal Document, Graudate School of Management, Rutgers University, NewYork, New Jersey, 1993.

[6] K. Böhm, A. Müller, and E. Neuhold, "Structured Document Handling - A Case for Integrating Databases and Information Retrieval," Proc. of CIKM '94, pp. 147-154, 1994.

[7] 김현기, 김연중, 이희주, 장재우, "하이퍼미디어 응용을 위한 구조 정보 관리 시스템의 구현," 한국정보과학회 논문지, 제3권, 제2호, pp. 127-138, 1997. 4.

[8] M. Tamer Özsu, Paul Iglinski, Duane Szafron, Sherine El-Medani, and Manuela Junghanns, "An Object-Oriented SGML/HyTime Compliant Multimedia Database Management System," ACM Multimedia 97, pp. 239-249, 1997.

[9] ISO, *International Standard ISO/IEC8879 :*

Information Processing - Standard Generalized Markup Language(SGML), Geneva/New York, 1986.

[10] 한에노, 박인호, 강현석, 김완석, "SGML 문서의 관리를 위한 객체지향 데이타베이스 설계," 한국정보처리학회 논문지, 제4권, 제3호, pp. 670-684, 1997. 3.

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

[12] 장원호, 임혜정, 박인호, 강현석, "HOMT : HyTime DTD 설계를 지원하기 위한 XOMT의 확장," 한국정보처리학회 논문지, 제5권, 제9호, pp. 2213-2223, 1998. 9.

[13] World Wide Web Consortium, [URL: http://www.w3.org/TR/1998/REC-xml-19980210]

박 인 호

1990년 경상대학교 컴퓨터과학과 (이학사)
1997년 경상대학교 컴퓨터과학과 (공학석사)
2000년 경상대학교 컴퓨터과학과 (공학박사)

관심분야 : 객체 지향 데이터베이스, SGML/HyTime/XML, 멀티미디어 데이타베이스

강 현 석

1981년 동국대학교 전자계산학과 (상학사)
1983년 서울대학교 계산통계학과 (이학석사)
1989년 서울대학교 계산통계학과 (이학박사)
1981년~1984년 한국전자통신연구원 연구원
1984년~1993년 전북대학교 전자계산학과 부교수
1993년~현재 경상대학교 컴퓨터과학과 교수

관심분야 : 객체 지향 데이터베이스, 멀티미디어 데이터베이스, 내장형 데이터베이스