論文 94-19-3-8

객체 지향 데이타베이스 기술을 결합한 향상된 하이퍼미디어 시스템의 설계 및 구현

正會員 李 圭 哲*

The Design and Implementation of an Enhanced Hypermedia System integrated with Oject-Oriented Database Technology

Kyu Chul Lee* *Regular Member*

이 論文은 1992年度 敎育部 지원 韓國學術振興財團의 自由公募課題 學術硏究助成費에 의하여 硏究되었음.

要 約

기존의 하이퍼미디어 시스템은 노드와 링크에 기반한 단순한 데이타 모델과 개개의 정보를 브라우징하거나 링크를 따라 항해하는 제한적인 정보 검색 능력을 제공한다. 본 논문에서는 이와 같은 하이퍼미디어 시스템에 객체 지향 데이타베이스 기술에서 지원하는 강력한 데이타 모델링 능력과 질의 기능을 결합하여 보다 향상된 하이퍼미디어 시스템인 AHEAD를 설계, 구현하였다. AHEAD에서는 객체 지향 데이타 모델링 기법을 이용하여 응용에 나타나는 데이타의 시맨틱 및 관계성 정보를 정확히 모델링하고, 노드와 링크에 대한 유용한 정보를 표현하며, 다매체 데이타의 구조 및 연산을 정의할 수 있게 한다. 또한 이들 모델링된 정보를 이용하여 다양한 형태의 효율적인 질의어 기능도 지원하고 있다.

ABSTRACT

Conventional hypermedia systems provide a simple data model based on nodes and link, and give us the limited capability in information retrieval such as browsing and navigating following the links. In this paper, we design and implement a hypermedia system (AHEAD) with enhanced data modeling and querying capability, which are strongly supported in object-oriented database technology. The AHEAD makes it possible to model the accurate data semantics and relationships in application domain, to represent useful information related to nodes and links, and to define the structure and manipulation operations of multimedia data. It also support the various querying capabilities by using modeling information.

*忠南大學校 컴퓨터工學科
論文番號 : 93175
接受日字 : 1993年 9月 14日

Ⅰ. 서 론

최근 들어 정보의 표현에 있어 기존의 순차적 형태를 탈피하고 인간의 연산작용과 비슷하게 비순차적으로 정보를 구성하며, 다매체 데이타를 처리할 수 있는 하이퍼미디어 시스템이 관심을 끌고 있다[3].

그러나, 하이퍼미디어에서는 단순하게 노드(node)와 이들을 연결시켜주는 링크(link) 구조에 기반을 두고 있기 때문에 응용(application)에서 필요한 데이타의 복잡한 구조나 관계성(relationship)을 모델링(modeling) 하는데 어려움이 있다. 또한 개개의 정보를 브라우징(browsing)하거나 링크를 통해 항해(navigation)하여 정보를 검색할 수 있을 뿐 어떤 조건을 만족하는 여러 개의 노드나 링크를 한꺼번에 검색할 수 있는 질의 기능은 하이퍼미디어 시스템에서 제공되지 않고 있다.

한편 하이퍼미디어에서는 텍스트, 그래픽, 음성, 화상 등 다양한 다매체(multimedia) 데이타를 포함할 수 있는데, 이러한 다매체 데이타에 대해 각각의 매체에 적합한 검색 및 조작등의 처리 연산을 지원하지 못하고 있는 실정이다[8][9].

본 논문의 목적은 노드와 링크에 기반한 단순한 구조와 제한된 정보 검색 능력을 지닌 현재의 하이퍼미디어 시스템에 객체 지향 데이타베이스 기술에서 지원하는 강력한 데이타 모델링 능력과 다매체 데이타 처리 및 질의 기능을 결합하여 보다 향상된 하이퍼미디어 시스템인 AHEAD(A Hypermedia System with Enhanced Data Modeling and Querying Capability)를 설계, 구현하는데 있다.

AHEAD에서는 객체 지향 모델링 기법[1][4]에 기반을 두어 노드와 링크에 대해 추가의 자세한 정보를 표현할 수 있도록 하이퍼미디어 객체(object)로 모델링하였으며, 응용에서 나타나는 데이타 시맨틱 및 관계성 정보를 객체 지향 개념의 일반화(generalization)와 집단화(aggregation) 등의 고수준의 추상화 기법을 이용해 모델링하였다. 또한, 각 매체 데이타의 정보 및 연산을 별도의 클라스(class)로 정의하여 효율적인 매체 정보의 관리 및 처리 연산을 제공하도록 하였다. AHEAD에서는 이와 같이 모델링된 정보들을 이용하여 응용 데이타는 물론, 노드와 링크 객체들의 정보를 효과적으로 검색할 수 있으며, 매체 데이타를 조작할 수 있는 질의어도 구현, 제공되고 있다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 하이퍼미디어 모델의 단점을 지적하고 본 논문에서 제안하는 하이퍼미디어 정보와 응용 데이타 및 매체 데이타 모델링에 대해 설명한다. 3장에서는 AHEAD 시스템에서 제공하는 질의 기능과 질의어 처리 과정 등을 제시하고, 4장에서는 AHEAD의 구조와 구현 내용을 기술한다. 5장에서는 본 논문과 관련된 기존의 하이퍼미디어 또는 다매체 데이타베이스 시스템 연구들을 비교 분석하였으며, 6장에서 결론을 맺는다.

Ⅱ. 하이퍼미디어 정보 및 응용 데이타 모델링

2.1 하이퍼미디어 데이타 모델의 단점

서론에서 언급하였듯이 기존의 하이퍼미디어 시스템은 단순하게 노드와 이를 비순차적으로 연결시켜주는 링크 구조에 기반을 두고 있다. 따라서 이를 이용하여 좀 더 복잡한 응용을 모델링하거나 다양한 다매체 연산 및 효율적 질의 기능 지원을 위해서는 다음과 같은 세가지 측면에서의 모델링 능력을 향상시킬 필요가 있다.

첫째는 응용에 나타나는 응용 데이타들에 대한 복잡한 구조 및 관계성을 효과적으로 모델링 하는 것이다. 하나의 예로, 학생에 대한 데이타를 모델링할 때, 각 학생은 오직 한 학과(DEPT)에 속하고, 자신의 주소로 시, 구, 동으로 복합된 데이타를 갖는다고 하자. 이를 하이퍼미디어에서 표현하려면 각 학생마다 일일이 관련된 학과와의 링크를 설정해야 함은 물론, 이때 학생은 반드시 오로지 한 학과에 소속된다는 제약성(constraints) 시맨틱을 표현할 수 있는 방법이 없다. 또한 주소는 시, 구, 동 등의 여러 요소가 모여 이루어지며, 이것이 학생의 주소로서 연관을 맺고 있다는 사실을 정확히 표현할 수가 없다. 이러한 문제의 해결을 위해, 본 논문에서는 응용에서 필요한 데이타 시맨틱을 객체 지향 데이타 모델링 기법으로 스키마를 정의할 수 있도록 제안하였다. 객체 지향 데이타 모델링 기법을 사용하면 위와 같은 학생 데이타에 대하여 객체 지시자(object identifier)를 이용한 관계성의 모델링 및 주소와 같은 복합 객체(complex object)의 모델링이 가능하다.

하이퍼미디어 데이타 모델의 시맨틱을 강화하는 두번째 측면은 노드와 링크 자체에 대한 정보 모델링이다. 하이퍼미디어 시스템에서는 사용자는 단지 노드들 사이에 한개 또는 그 이상의 링크들이 있다는 것만을 알 뿐이며, 이것을 유사한 타입의 모든 노드들 사이에서 유지될 수 있는 관계성(relationship)으

로 일반화 시키는 것은 어렵다. 이 모델에서 상위 계층 노드(high-level node)는 단지 기본 노드(elementary node)들의 연관 구조만을 표현할 뿐 그들의 구조가 변경될 때 이를 능동적으로 반영하지 못하고, 또한 노드들의 구조(structure)와 계층적 관계도 나타내지 못하고 있다[13].

이러한 문제점이 발생하는 주요 원인은 하이퍼미디어 모델이 충분한 시맨틱을 제공하지 못한다는 것이다. 데이타 모델링 기법을 노드와 링크에 이용한다면, 노드와 링크가 가질 수 있는 추가 정보에 대하여 모델링할 수 있다. 따라서 비슷한 타입을 갖는 노드들에 대해서도 그 애트리뷰트 형태로 상세한 정보를 모델링할 수 있으며 링크에 대해서는 연관된 노드들의 정보와 관계성도 나타낼 수 있다. 또한 노드들이 링크되어 있는 상태 정보도 쉽게 파악할 수 있으므로 노드들의 구조 정보도 쉽게 표현할 수 있다.

셋째는 매체 데이타에 대한 모델링 측면이다. 하이퍼미디어 시스템에서의 매체 데이타는 수동적(passive)인 형태로 단지 데이타만을 저장하는 수단이며, 매체를 다루기 위한 내부 구조 및 연산에 대한 정보를 표현하는데 미흡하다. 본 논문에서는 객체 지향 모델에서 메소드(method)를 이용해 연산을 모델링하는 기법을 이용하여 이를 지원한다. 또한 매체 데이타의 특성을 같이 모델링하여 매체에 대한 질의도 가능하게 한다.

2.2 AHEAD에서 응용 데이타에 대한 모델링

AHEAD에서 응용 데이타들은 객체 지향 모델의 클라스로 모델링된다. 따라서 하나의 응용에 관련된 클라스들은 모여 스키마(schema)를 구축하며, 이 스키마는 선언된 ISA 관계성에 따라 일반화 계층(generalization hierarchy)를 구성하여 특성 계승(property inheritance)을 지원하며, 또한 집단화 계층(aggregation hierarchy)를 구성하여 복잡한 구조의 복합 객체(complex object) 모델링 기능을 지원한다.

예로, 대학에서의 응용을 위한 데이타 중 학생에 대한 클라스는 AHEAD에서 다음과 같이 선언된다.

```
CLASS Student ISA Person
ATTRIBUTES
    student_no      : int ;
    dept            : Dept ;
    courses         : SET OF Course ;
    addr            : TUPLE(dong : string ; gu :
                      string ; city : string) ;
END
```

Student 클라스는 Person의 서브클라스로 선언되며 5개의 애트리뷰트를 갖는다. 각 애트리뷰트는 자신의 도메인(domain)으로 다른 클라스를 가지며, 이때 도메인은 기본적인 타입인 int, string, float 등을 가질 수 있고, 또한 사용자가 정의한 클라스 또는 이들의 복합 객체를 가질 수 있다. 예를 들어, dept 애트리뷰트는 사용자 정의 클라스인 Dept를 도메인으로 가지므로, 이 클라스와 관계성을 갖는 것을 모델링할 수 있으며, courses인 경우에는 여러개의 Course와 관계를 맺고 있다는 제약성(constraints)을 표현한다. 한편 애트리뷰트 addr은 투플(tuple) 타입 구성자로 선언된 복합 객체를 자신의 도메인으로 취하며, Image 타입의 다매체 데이타도 picture 애트리뷰트의 도메인으로 사용될 수 있다.

만일 이와 같은 정보를 하이퍼미디어에서 표현하려 하면, 각 학생 개개인의 데이타에 대해 관련된 학과(Dept), 과목(Course)등을 일일히 링크로 연결해야함은 물론 addr에 대한 정보는 묶어서 표현하기 어려우므로 중간 노드를 하나 더 두어 표현할 수 밖에 없으며, Student 전체를 대상으로 하는 "DB 과목을 수강하는 학생들의 학과를 모두 검색하라"라는 식의 질의는 매우 처리하기가 힘들어진다. 따라서 이와 같은 데이타 모델링 능력을 향상시키는 것은 응용 데이타의 구조나 관계성을 표현하고 질의 또는 처리하는데 매우 중요하다. 그러나 이들 모델링된 정보를 유용하게 활용하려면, 시스템이 이를 효율적으로 저장, 지원해야 하는데 이 목적을 달성하기 위하여 일반 데이타베이스 시스템에서의 카다로그(catalog)와 유사하게 AHEAD에서는 ClassInfo와 Attr이라는 두개의 시스템 지원 클라스를 제공한다. 여기서 중요한 것은 응용 클라스에 대한 정보를 관리하는 ClassInfo와 Attr도 응용 데이타와 마찬가지로 객체 지향 모델의 클라스로서 표현된다는 것이다.

그림 1에 클라스 ClassInfo와 Attr의 구조 및 제공되는 메소드(method)들을 도시하였다. 클라스 ClassInfo와 Attr의 수퍼 클라스인 SchemaInfo는 이들을 효율적으로 관리하기 위하여 정의된 추상 클라스이다. 클라스의 정의는 구현에 사용된 Objective-C의

문법에 따랐다.

ClassInfo는 응용을 구성하는 클라스의 이름, 애트리뷰트들의 식별자, 각 클라스의 수퍼클라스 및 서브클라스등의 메타정보(meta information)를 관리한다. Attr은 각 클라스내에서 정의된 애트리뷰트들에 대한 정보를 관리하며, 여기에는 애트리뷰트의 이름, 애트리뷰트가 속한 클라스의 이름, 애트리뷰트의 타입, 만일 애트리뷰트가 상속된 것일 때 이 애트리뷰트가 처음 정의된 수퍼클라스의 이름, 그리고 응용 데이타에서의 애트리뷰트의 위치 정보등을 관리한다. 여기에서 애트리뷰트의 타입은 정수(integer), 실수(float), 비정형 데이타 타입인 텍스트(text), 이미지(image) 그리고 사운드(sound)등의 타입을 가질 수 있다.

그림 1에서 나열된 각 메소들은 AHEAD의 다른 시스템 요소(component)에서 클라스와 애트리뷰트 정보를 사용하기 위해서 호출하여 사용하는 함수들이다.

```
CLASS ClassInfo ISA SchemaInfo
{
    char  *className;          /* 클라스 이름 */
    int   current_inst_num;    /* 생성시 부여되는 인스탄스 번호 */
    Attr  *attribute[];        /* 소유한 애트리뷰트들의 식별자 */
    ClassInfo  *superClass;    /* 수퍼 클라스 식별자 */
    ClassInfo  *subClass[];    /* 서브 클라스들의 식별자 */
}
- (char*)getMyClassName;
- (int) getCurrentInstNumber;
- (char*)getSuperClass: (char *)aClassName;
- (char*)getSubClass:  (char*)aClassName;
- (char*)getAttr: (char*)aClassName;


CLASS Attr ISA SchemaInfo
{
    char  *attrName;           /* 애트리뷰트 이름 */
    ClassInfo  *ownClass;      /* 소유자 클라스 식별자 */
    ClassInfo  *inheritedClass;  /* 상속받은 클라스 식별자 */
    type attrType;             /* 애트리뷰트 타입 */
    rect  *attrPosition;       /* 애트리뷰트가 표시될 위치 정보 */
}
- (char*)getInheritedClass: (char*)attrName;
- (char*)getType: (char*)attrType;
- (char*)getAttrNameForPoint: (point)aPosition
                        at: (char*)aClassName;
- (rect)getAttrRectForAttrName: (char*)attrName
                        at: (char*)aClassName;
```

그림 1. 스키마 관리를 위한 ClassInfo과 Attr 클라스

Fig 1. ClasInfor and Attr classes for shcema management

이러한 정의를 예를 들어 설명하기 위하여 하나의 응용에서 GRADUATE라는 클라스를 하나 생성한다고 가정하자. 클라스 GRADUATE가 major(전공), prof(지도교수) 등의 2개의 애트리뷰트를 가진다고 가정하면, 생성시 클라스 ClassInfo는 다음과 같은 인스탄스를 하나 생성하게 된다.

```
(GRADUATE,        /* 클라스 이름 */
1,                /* 인스탄스 번호. 첫 인스탄스
                     라 가정 */
{major, prof},    /* 애트리뷰트들 */
Student,          /* 수퍼클라스. 수퍼클라스가
                     Student라 가정. */
NULL)             /* 서브클라스는 없슴. */
```

이상에서 설명하였듯이 클라스 ClassInfo와 Attr는 응용을 구성하는 클라스에 대한 클라스 자체의 정보와 클라스를 구성하는 애트리뷰트들에 대한 정보를 관리하며, 이러한 모델링은 응용 데이타에 대한 체계적인 관리를 가능하게 한다.

2.3 AHEAD에서 노드와 링크 정보의 모델링

노드 및 링크는 하이퍼미디어 시스템의 가장 기본적인 구성 요소이다. AHEAD에서는 하나의 객체의 어떤 일부분도 하이퍼미디어의 앵커 노드가 될 수 있도록 허용하며, 하나의 클라스에 속한 각 객체들은 동적(dynamic)으로 다른 어떤 객체와도 링크로 연결될 수 있다. AHEAD에서는 이와 같은 동적인 노드와 링크 정보를 보다 효율적으로 관리하기 위해 노드와 링크 정보를 따로 관리하고 있는데 이를 지원하는 것이 바로 추상 클라스(abstract class)인 Node_Link_Manager이다.

AHEAD는 링크의 타입으로 참조 링크(referential link)와 부연 링크(annotation link)[2]를 제공하도록 설계되었다. 하나의 노드에 대하여 여러개의 참조 링크와 하나의 부연 링크를 설정할 수 있는데, AHEAD에서는 링크 설명 도구(Link_Explaination)를 제공함으로써 여러개의 링크가 설정되어 있을 때 그 중의 하나를 선택할 수 있도록 정보를 기록할 수 있다.

링크를 생성하기 위해서는 선택한 인스탄스에서 앵커(anchor) 노드를 먼저 설정한 후에 원하는 목적지를 선택한다. 이때 링크 타입과 링크에 대한 설명을 기술할 수 있는 도구가 제공되는데, 그림 2의 링크 설명 도구가 바로 그것이다. 이 도구는 링크의 생성이 완료될 때 나타나며 Destination 부분은 시스템이 자동으로 가입해 주도록 설계 하였다. 위예에서는

BOOK이라는 클라스의 2번 인스탄스가 목적지임을 나타낸다.

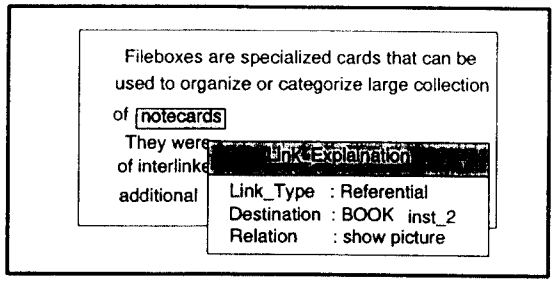


그림 2. 링크 설명 도구
Fig 2. Tool for link explanation

하나의 노드에는 여러개의 링크가 존재할 수 있는데 linkPath 기능을 선택하면 그림 3과 같은 링크 안내자(link guide)가 나타나게 되며 여기에 있는 여러 링크 중에서 원하는 것을 선택하면 그 링크의 목적지로 항해할 수 있다.

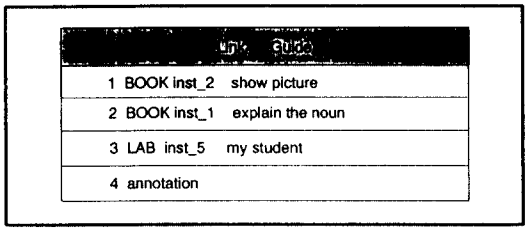


그림 3. 링크 안내자
Fig 3. Link guide

노드와 링크에 관한 정보들은 Node_Link_Manager라는 클라스의 서브 클라스로 정의된 NODE, RefLink 및 AnnoLink등 3개의 클라스에 의해 관리된다. 다음은 클라스 NODE의 구조와 메소드를 보여주고 있다.

```
CLASS NODE ISA Node_Link_Manager
{
ClassInfo *ownClass ;   /* 소유자 클라스 식별자 */
int instance_num ;      /* 인스탄스 번호 */
Attr *anchor ;          /* 앵커 노드 식별자 */
char *kind :            /* 노드 종류 */
RefLink *referential[ ] ;   /* 참조 링크로 식별자 */
AnnoLink *annotation[ ] ;   /* 부연 링크 식별자 */
}

- node_kind : aNode ;
- link_kind : aLink ;
- link_type : aType ;
```

이 정의에서도 알 수 있듯이 클라스 NODE는 오직 하나의 앵커와 두 종류의 링크를 애트리뷰트로 갖는다. 클라스 NODE는 특정 종류의 노드와 링크를 추출하는 node kind와 link kind, 타입별 링크를 추출하는 link_type 등의 메소드를 제공한다. 아래에 링크에 부가적 타입에 대한 정의와 메소드를 보인다.

```
CLASS RefLink ISA Node_Manager
{
NODE *anchor ;      /* 앵커 노드 식별자 */
NODE *toClass ;     /* 목적지 노드의 클라스 */
int instance_num ;  /* 목적지 노드의 인스탄스 번호 */
Attr *destAttr ;    /* 목적지 노드 식별자 */
Text linkExplain ;  /* 링크 설명자 */
char *kind          /* 링크 종류 */
}
- source : aNode ;
- destination : aNode ;
- follow : aRefLink ;
- back : aRefLink ;

CLASS AnnoLink ISA Node_Link_Manager
{
NODE *anchor ;      /* 링크 노드 */
Text *annotate ;    /* 표시될 주석 */
}
```

클라스 RefLink는 앵커 노드, 목적지 노드가 속한 클라스 및 인스탄스 번호, 목적지 노드 식별자와 링크 설명자등의 애트리뷰트를 갖는다. 클라스 RefLink는 네개의 메소드를 가지며, 이들은 앵커 노드를 얻기 위한 source와 이동할 참조 노드를 선택하는 destination, 링크를 따라 항해하는 follow와 역추적하는 back이다.

부연 링크로 앵커와 연결되어 지는 부연 노드는 단

순히 앵커 노드에 대한 부가적인 설명만을 위한 것이기 때문에 목적지가 필요하지 않다. AHEAD에서는 부연 노드를 텍스트로 표현되는 하나의 화일로 다루고 있다.

이렇게 노드와 링크를 모델링 함으로써, 인지적 부담을 완화 시킬 수 있는 링크 안내자와 같은 도구에 관련된 정보를 나타낼 수 있으며, 또한 노드간의 시맨틱 관계성도 표현할 수 있다. 응용의 모델링에서와 마찬가지로 이들에 대한 모델링 정보는 질의할 수 있는 환경을 제공한다.

2.4 매체 정보의 모델링

본 연구에서는 AHEAD의 구현 환경에서 사용 가능한 매체인 텍스트, 이미지, 그래픽, 오디오를 그 대상으로 하였다[15][16][17]. AHEAD에서는 각 매체들을 클라스로 분류하여 정의하고, 각 매체에 종속되어 연산을 수행하는 특징을 지닌 연산자들 즉, 매체 관련 연산자들을 추출하여 메소드로 정의하였다. AHEAD에서의 매체 정보 모델링의 중요한 점은 매체 정보의 효율적인 이용에 필요한 매체의 구조 정보 및 연산자들이 모두 클라스로 모델링되어 시스템에서 제공되므로 어떤 매체로 표현된 데이타의 조작이 쉬워진다는 점이다. 예를 들어, 하이퍼미디어 안에 있는 모든 텍스트 데이타 중에서 어떤 특정 키워드를 가진 데이타를 검색하는 기능을 매체 자체에 메소드로 구현하여(즉, Text 타입안에 contatins란 메소드) 지원할 수 있다는 것이다.

본 절에서 추출한 연산자들의 구현 내용은 복잡하여 그 나열을 피하고, 다음과 같은 표기법으로 정의하였다. 예를 들어 "contains : Text × String → Boolean" 이라는 표기는 텍스트 타입과 스트링 타입을 가진 객체를 인자로 하여 contains 연산을 수행하면 그 결과가 boolean 값이 된다는 의미이다. 본 논문에서는 지면상 대표적 매체인 텍스트와 이미지에 대한 정의만을 보이도록하며, 자세한 내용은 [17]을 참조하기 바란다.

2.4.1 텍스트

텍스트는 연속적인 문자 스트링의 집합으로서 실제 데이타를 관리하는 contents, 데이타가 표현될 실제 프레임의 크기(높이, 너비)를 가지고 있는 size, 텍스트의 데이타 형식에 대한 format, 그리고 해당 텍스트의 keyword등을 애트리뷰트로 가진다. 텍스트 클라스의 정의는 다음과 같다.

```
CREATE CLASS Text ISA Media
ATTRIBUTES
    contains : SET OF String ;
    size     : TUPLE(height : int, width : int) ;
    format   : CHOICE(ASCII, EBCDIC) ;
    keyword : String ;
END

contains : Text × String → Boolean.
append : Text × Block → Text.
delete : Text × Block → Text.
equal : Text × Text → Boolean.
keyword : Text → String.
```

텍스트에 대한 매체 관련 연산자에는 특정 문자열을 가지고 있는가를 체크하는 contains, 텍스트에 새로운 문단을 삽입하는 append, 텍스트에서 특정 문단을 삭제하는 delete, 두 텍스트가 같은 가를 체크하는 equal, 그리고 텍스트의 키워드를 찾아내는 keyword 연산등이 있다.

2.4.2 이미지

이미지는 실제 데이타에 대한 객체 식별자(object identifier : OID)를 관리하는 content, 데이타가 표현될 실제 프레임의 크기(높이, 너비)를 가지고 있는 size, 그리고 이미지의 데이타 형식에 대한 format등을 애트리뷰트로 가진다. 이미지 클라스의 정의는 다음과 같다.

```
CREATE CLASS Image ISA Media
ATTRIBUTES
    contents : SET OF Object ;
    size     : TUPLE(height : int, width : int) ;
    format   : CHOICE(TIFF, GIF, PCX) ;
END

zoomIn : Image × Ratio → Image.
zoomOut : Image × Ratio → Image.
rotate : Image × Angle → Image.
shift : Image × Direction → Image.
equal : Image × Image → Boolean.
superimpose : Image × Image → Boolean.
colorTransform : Image × Color → Image.
imageConversion : Image × Format → Image.
```

union : Image × Image → Image.
intersect : image × image → Image.

이미지 관련 연산자에는 이미지를 확대하고 축소하는 zoomIn과 zoomOut, 이미지를 회전 시키는 rotate, 특정 방향으로 이동시키는 shift, 두 이미지가 같은가를 체크하는 equal, 두 이미지를 겹치게 하는 superimpose, 이미지의 색을 변환시키는 colorTransform, 이미지 저장 형식을 변환 시키는 imageConversion, 그리고 이미지의 union과 intersect등이 있다.

Ⅲ. 질의 기능의 향상

AHEAD에서는 모델링된 하이퍼미디어, 응용 데이타, 매체 정보들을 이용하여 응용 데이타는 물론 노드와 링크 객체들의 정보를 효과적으로 검색할 수 있으며, 매체 데이타를 조작할 수 있는 질의어를 구현, 제공하고 있다. 본 장에서는 기존의 하이퍼미디어 시스템에서는 지원되지 않았던 이러한 질의어 기능들이 필요하게 된 배경과 AHEAD에서 제공하는 질의어 유형 및 질의어 처리기의 구현에 대하여 설명한다.

3.1 질의 기능의 필요성

하이퍼미디어에서는 링크를 통해 개개의 정보를 브라우징하거나 항해하여 정보를 검색할 수 있도록 하고 있으며 이러한 과정은 모두 사용자의 흥미도에 따라 서로 다르게 진행될 수 있다. 이러한 방법은 효율적인 데이타 검색의 측면에서 볼 때 많은 문제점[5]을 안고 있다. 즉, 저장된 데이타량이 매우 많거나 하이퍼미디어 네트워크의 구조를 잘 이해하지 못한 사용자가 정보를 검색할 경우에 원하는 정보를 브라우징하거나 항해하려면 많은 어려움이 뒤따르게 된다. 예를 들면, 10만개의 노드로 구성된 정보가 있다고 해도 브라우징을 통해 원하는 노드로 찾아갈 수 있는 능력은 제한적이라 할 수 있다. 또 다른 문제점으로는 어떤 조건을 만족하는 여러개의 노드나 링크를 한꺼번에 집합(set) 단위로 검색할 수 있는 방법이 제공되지 않는 것이다.

이러한 이유 때문에 하이퍼미디어에서 제공하는 브라우징 및 항해 기능과 데이타베이스에서 제공하는 질의어 기능을 자유롭게(seamless) 제공할 수 있는 시스템의 필요성이 대두되고 있다. 한편, 하이퍼미디어에 이러한 질의어 기능을 통합함은 물론, 특히 링크의 속성에 따라 질의할 내용의 범위(range)를 확장할 수 있으며, 질의 구성시에 구조적인 정보도 함께 포함할 수 있는 추가의 기능도 제공할 수 있다. 또한 매체 데이타를 조작할 수 있는 질의어의 표현도 쉽게 결합될 수 있다.

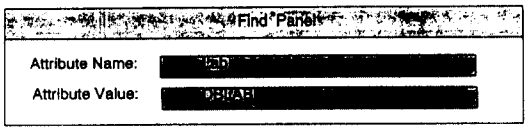
3.2 질의어 유형

본 시스템에서는 질의 기능을 향상시키기 위해 크게 4가지 유형의 질의어를 고려하였다. 첫째, 브라우징 및 항해를 이용한 질의, 둘째, 객체 지향 질의 대수 연산자를 이용한 질의, 세째, 하이퍼미디어 연산자를 이용한 질의 그리고 마지막으로는 매체 관련 연산자를 이용한 질의이다. AHEAD에서의 상세한 질의어 구문과 질의어 처리기 구현의 자세한 부분은 [17]에 기술하였다.

3.2.1 브라우징 및 항해를 이용한 질의

AHEAD에서는 브라우징 및 항해도 하나의 질의 형태로 간주한다. 즉, 사용자는 질의를 한 다음 검색되어 나온 결과들을 일일히 브라우징할 수도 있고, 또한 직접 브라우징을 수행할 수도 있으며, 노드에 설정된 링크를 따라 항해할 수도 있는데, 이러한 과정은 모두 하나의 질의 형태[16]이다. 따라서, 뒤의 4장에서 설명될 브라우저 메뉴에서 제공되는 first, next, last, previous, find 기능과 링크 메뉴에서 제공되는 follow와 back 기능 등을 선택하는 사용자의 행위는 질의의 한 형태로 취급된다. 브라우저를 메뉴를 통해 정보를 검색하는 과정은 다음과 같다. 먼저 클라스 메뉴의 'use' 기능을 이용해 원하는 클라스를 선택하면, 선택된 클라스의 해당 인스탄스를 브라우징해서 볼 수 있다. 이때 검색된 여러개의 인스탄스들은 각각 first, next, last, previous 기능을 이용하여 원하는 방향으로 이동될 수 있다. 그리고, 이러한 인스탄스들 중에서 어느 특정한 애트리뷰트 값만을 만족하는 인스탄스를 브라우징해서 보고자 할때는 "find" 기능을 이용한다. 예를 들어, 'find' 기능이 선택되면 다음 같은 패널이 화면에 나타나는데, 이때 Student 클라스의 애트리뷰트 이름이 "Lab"이고, 그 값이 "DBLAB"인 조건을 설정하면 검색된 Student 클라스의 모든 인스탄스 중에서 "DBLAB" 소속인 학생들만 다시 검색될 것이다.

항해를 이용한 질의는 이미 설정된 링크가 존재할 때 가능하다. 예를 들면, 브라우징을 통해 검색된 인스탄스에 링크 설정 노드가 나타나면 "follow" 기능



을 이용하여 다음 목표 노드를 찾을 수 있으며, "back" 기능을 이용하여 지금까지 항해한 근원 노드들을 역추적할 수 있다.

3.2.2 객체 지향 질의 대수를 이용한 질의

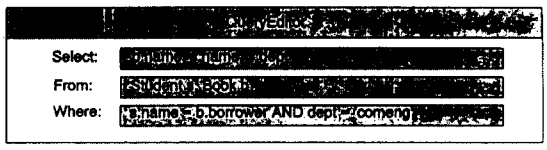
AHEAD에서는 기본적으로 모델링된 데이타 모두가 객체 단위로 저장되므로 이를 한꺼번에 집합 단위로 검색할 수 있는 질의 기능이 필요하다. 기존의 관계 DBMS에서의 SQL과 같은 데이타베이스 언어는 집합 단위의 연산을 지원하며, 이는 내부적으로는 관계 대수로 변환되어 처리된다.

AHEAD에서의 질의어는 SQL 형태의 select-from-where 절 형태를 띄고 있으나, 이는 질의어 처리기에서 분석되어 기본적인 객체 지향 대수로 변환되어 처리되어 진다.

본 논문에서는 [10]에서 제시한 객체 지향 대수의 중요 연산들을 구현[17]하였으며, 그 내용은 다음과 같다.

- select : 주어진 조건을 만족하는 객체를 검색
- Image : 주어진 객체의 특정 애트리뷰트 하나를 검색
- project : 주어진 객체의 여러 애트리뷰트를 검색
- Ojoin : 객체간의 관계성을 찾아 검색
- 집합 연산 : Union, Difference, Intersect 등 객체 집합간의 연산

다음은 "책을 대출한 컴퓨터공학과 학생의 이름, 학과, 책 이름을 검색하라"는 질의 예이며, 이때 질의 내의 s와 b는 Student와 Book 클라스의 객체 변수이다. 이 질의는 시스템 내부에서 select, Ojoin, project 연산으로 변환되어 수행되게 된다.

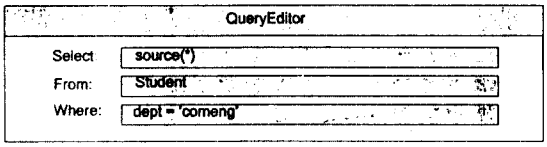


3.2.3 하이퍼미디어 연산자를 사용한 질의

하이퍼미디어 연산자는 2.3절에서 모델링된 노드와 링크 정보를 기반으로 하고 있으며, 질의 내에 사용할 수 있는 하이퍼미디어 연산자들은 하이퍼미디어 정보를 담고 있는 클라스인 NODE와 RefLink, AnnoLink에 정의된 메소드들이 된다. 즉, 다음과 같은 연산자를 사용하여 집합 단위로 노드 및 링크 정보를 검색할 수 있다.

- node_kind : 특정 종류의 노드 모두를 검색
- link_kind : 특정 종류의 링크 모두를 검색
- link_type : 참조 또는 부연 링크등의 타입에 따른 검색
- source : 참조 링크의 앵커 노드를 모두 검색
- destination : 참조 링크로 연결된 목표 노드를 모두 검색

이중 특히 source와 destination 연산은 질의어의 target 절 즉, select 절에서 사용 가능하다. 이러한 예는 다음 질의에서 볼 수 있다. 다음은 "학생들 중에서 컴퓨터공학과 소속인 학생의 모든 근원 노드를 검색하라"는 질의 표현이다.



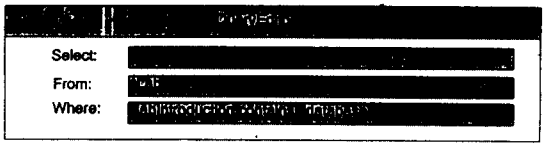
위의 질의어를 수행하는 절차는 먼저 From 절에서 기술된 Student 클라스의 인스탄스중 Where 절의 조건에 맞는 인스탄스들만을 추출하고, 이들을 인자(argument)로 하여 메타 정보로 저장된 RefLink 클라스의 source 메소드를 수행하게 된다. 그 결과들은 브라우저를 통해 화면에 프리젠테이션(presentation)되고 사용자는 자신의 요구에 따라 하나씩 브라우즈할 수 있다.

3.2.4 매체 관련 연산자를 사용한 질의

기존의 하이퍼미디어에서는 텍스트, 그래픽, 음성, 화상 등 다양한 다매체 데이타를 포함할 수는 있으나 이러한 다매체 데이타에 대해 각각의 적합한 검색 및 조작등의 처리 연산을 지원하지 못하고 있다.

AHEAD에서는 이러한 다매체 데이타를 처리할 수 있는 연산자를 지원하고 있다. 이러한 매체 관련 연산자는 2.4절 매체 정보의 모델링에서 정의한 각 매체 클라스의 메소드들이 된다.

예를 들어, 클라스 Lab의 하나의 애트리뷰트인 labIntroduction이 2.4.1절에 모델링된 Text 타입으로 선언되었다면, Text타입에 정의된 contains란 메소드를 이용해 다음과 같은 질의를 할 수 있다.



3.3 질의어 처리기의 구현

질의어 유형에서 기술했던 질의어들은 특정 객체들의 콜렉션들 중에서 조건을 만족하는 객체들을 추출하는 형태를 띠고 있다. AHEAD에서 제공하는 질의어 처리 과정은 다음 그림 4에서 보듯이 질의 생성기부터 어휘 분석기, 구문 분석기, 질의 실행 처리기까지 크게 4단계로 구분할 수 있으며, 질의 방법은 직접 사용자가 질의를 작성하는 방법과 브라우저를 사용하여 데이타를 브라우징하는 방법으로 나눌 수 있다.

질의 생성기는 질의를 작성하는 단계이며, AHEAD에서는 질의 편집기(Query Editor)를 이용하여 질의를 작성할 수 있도록 제공하고 있다. 질의를 작성할 때, 각 객체에 정의된 메소드 및 애트리뷰트들은 정보 검색시에 연산자나 비교 대상으로 사용된다.

어휘 분석기는 질의를 받아들여 일련의 토큰(token)을 생성하며, 이들을 이용하여 질의 구조체를 생성한다. 즉, 질의를 각각 Select, From, Where 절로 구분하고, 각 절들을 토큰들의 모임으로 구성한 질의 구조체를 생성한다.

구문 분석기는 질의 구조체에 나열된 토큰들의 타입을 조사하여 질의가 구문에 맞는지를 검사하며, 구문에 맞으면 질의 실행 처리 과정이 진행되고 만약 그렇지 않으면, 에러 메세지를 발생시킨다.

질의 실행 처리기는 어휘 분석과 구문 분석 과정을 수행한 타당한 질의를 실제 실행시킨다. 이때, 질의 처리 순서는 from, where, select 순으로 실행된다.

예를 들어, 3.2.4절의 질의는 어휘 분석과 구문 분석 과정을 거쳐 다음과 같이 실행되게 된다. 먼저,

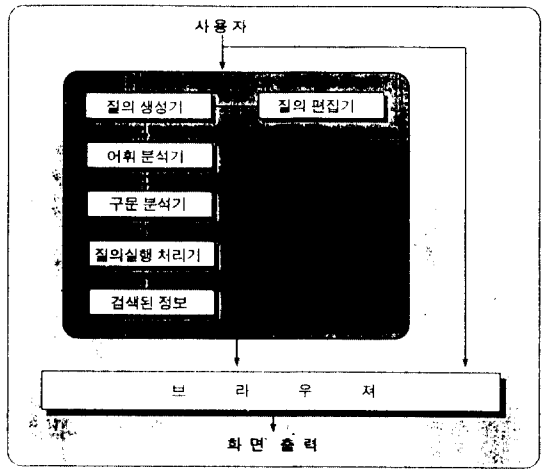


그림 4. 질의어 처리 과정
Fig 4. Query processing procedure

from절에서 클라스 Lab을 선택하여 애트리뷰트 labIntroduction이 이 클라스에 존재하는지를 조사하는데, 이 애트리뷰트의 존재 여부는 ClassInfo라는 스키마 정보 클라스를 참조하여 조사한다. 만약 그러한 애트리뷰트가 존재하면, 다시 Attr이라는 클라스를 참조하여 그 애트리뷰트의 타입을 조사한다. 이때, labIntroduction이 Text 타입으로 선언되어 있으므로 Text 클라스에 연산자 contains가 있는지를 검사하여 만약 그러한 연산자가 있으면, contains : 'database'라는 메세지를 실행한 후 조건에 맞는 인스턴스들을 추출한다. 마지막으로, 객체 지향 대수 select에 해당하는 프로시쥬어를 실행시켜 조건에 맞는 그 결과들을 출력한다.

본 논문에서 구현한 질의어 처리기에는 질의어 최적화(query optimizaiton) 부분은 구현되지 않았다. 따라서 위에서 설명한 바와 같이 주어진 질의어를 분석한 뒤 관련된 클라스에 정의된 메소드를 부르는 일련의 메세지들로 변환하여 수행하도록 하는 프로토타입으로 구현되었다. 본 프로토타입의 구현으로 객체 지향 기법으로 모델링된 하이퍼미디어 및 응용 데이타, 매체 관련 정보를 이용하여 충분히 효과적인 질의 기능이 지원이 가능함을 확인할 수 있었으나, 궁극적으로 완전한 질의 처리기를 구현하기 위해서 필요한 질의어 최적화에 대하여는 추후에 좀 더 심도 있는 연구가 필요하다.

한편, 기존의 [5]에서 제시된 대규모 하이퍼베이스

에서의 질의 기능 연구와 본 AHEAD에서의 질의기능 지원의 차이점은, [5]에서는 조건을 만족하는 노드나 링크를 집합 단위로 검색하는 하이퍼미디어 정보에 대한 질의 기능만을 제공하는데 반해, AHEAD에서는 객체 지향 대수를 이용한 일반적 데이타베이스 질의 기능과 매체에 정의된 메소드를 이용한 질의 등 좀 더 다양한 강화된 질의 기능을 지원한다는데 있으며, 또한 이 모든 것이 단일하게 객체 지향 개념을 기반으로 하고 있다는 점이 특징이다.

Ⅳ. AHEAD 시스템의 구현

4.1 구현 환경

본 시스템은 NeXT 스테이션에서 Objective-C 언어를 사용하여 구현하였다.

NeXT 스테이션에서는 응용 프로그램을 지원하는 APP KIT나 사운드의 조작을 쉽게 할 수 있는 SOUND KIT을 제공하며, 사용자에게 원하는 GUI(Graphic User Interface) 환경들을 쉽게 구현할 수 있도록 도와 주는 Interface Builder를 제공한다. 사용자는 응용에 표현하고자 하는 윈도우나 메뉴 바(menu bar), 버튼(button), 패인(pane), 스크롤링(scrolling)을 지원하는 슬라이더(slider) 등을 팔레트(palette)에서 단지 마우스 드래깅하여 사용할 수 있으며 이렇게 구성된 사용자 인터페이스는 unparse를 통해 직접 Objective-C 코드(code)로 생성 된다. 아이콘이나 버튼을 마우스 클릭할 때 발생할 수 있는 사건(event)들을 처리하기 위한 구현도 간단하다. 아이콘 화일 박스에서 제공하는 여러 가지 형태의 아이콘들을 사용자 윈도우로 가져다 놓고 링크과정과 그 링크에 대한 메소드 정의를 수행함으로써 가능하다. 이렇게 구현된 인터페이스도 unparse를 통해 Objective-C 언어로 코드화될 수 있으며 사용자는 여기에 더 필요한 코드들을 자세히 기술하면 된다.

Objective-C 언어는 메소드의 구현을 포함하는 .m 화일 및 클라스의 인스탄스와 메소드 명세(specification)를 정의한 헤더(header) 화일 즉, .h 화일을 기본 프로그램을 구성하고 있다. 사용자는 주 화일에서 프로그램의 전체적인 흐름을 제어하며, 각 클라스의 인스탄스와 그 클라스에 정의할 수 있는 클라스 메소드 및 인스탄스 메소드들은 헤더 화일에 정의할 수 있는데 이 메소드들은 .m 화일로 구현(implementation)된다.

NeXT에서 Objective-C 언어로 구현된 응용프로그램은 객체를 생성시키고 저장하지 않으면 해당 응용 프로그램이 실행을 마치게 될 때 그 값을 잃어버리게 된다. 따라서 응용 프로그램이 실행을 마치기 전에 모든 값들을 반드시 기억 장소에 저장해야만 한다. AHEAD에서는 이러한 일들을 저장 관리기가 맡아 운영하도록 구현했으며 영속적(persistent)인 데이타 저장 장소를 관리한다.

4.2 시스템 메뉴의 구성

AHEAD의 전체 메뉴 계층도를 나타내면 다음 그림 5와 같다. 주 메뉴로는 클라스의 생성을 처리하는 Class, 다매체 데이타의 편집 기능을 제공하는 Edit, 질의 처리를 위한 Query, 인스탄스를 브라우징하기 위한 Browse, 링크 연산을 수행하는 Link, 그리고 시스템 수행을 종료하기 위한 Quit가 있다.

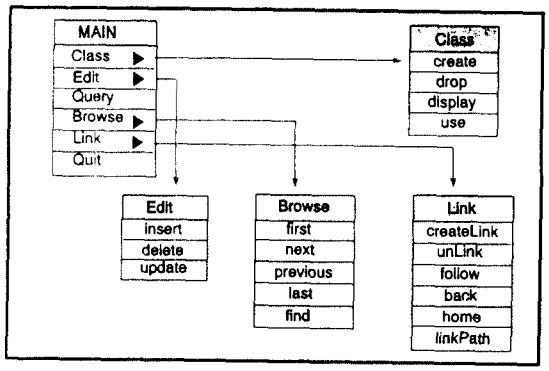


그림 5. 시스템 메뉴 계층도
Fig 5. System menu hierarchy

Class 메뉴에서는 각 클라스를 생성(create) 및 삭제(drop)하는 기능과 클라스의 논리 정보와 배치 정보를 보여 주는(display) 기능, 그리고 클라스를 열어주는(use) 기능이 있다. Edit 메뉴에서는 인스탄스의 실제 값을 처리하게 되는데 데이타를 삽입(insert), 삭제(delete), 갱신(update)하는 기능이 있다. Query 메뉴에서는 질의 편집기(query editor)를 제공한다. Browse 메뉴에서는 한 클라스의 첫번째 인스탄스를 보여 주는(first) 기능, 현재 화면에 나타난 인스탄스의 다음 인스탄스를 보여 주는(next) 기능, 현재 인스탄스 이전의 인스탄스를 보여 주는(previous) 기능, 마지막 인스탄스를 보여 주는(last) 기능, 그리고 어떤 조건에 의해 원하는 인스탄스를 찾는(find) 기

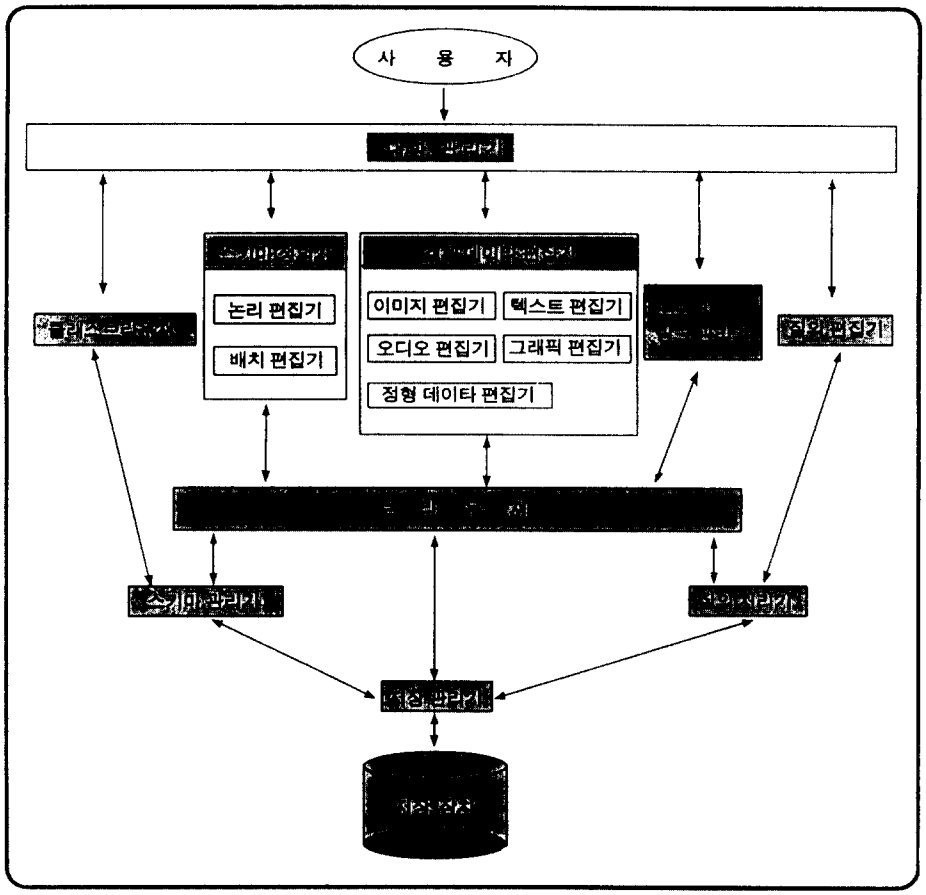


그림 6. AHEAD의 구성도.
Fig 6. Configuation of AHEAD

능이 있다. Link 메뉴에서는 링크를 생성(createLink), 및 삭제(unLink)하는 기능과 링크를 따라 목적지로 항해하는(follow) 기능, 따라온 링크를 다시 역 추적하는(back) 기능, 항해중에 맨 처음의 인스턴스로 돌아가는(home) 기능, 링크 정보를 제공하는(linkPath) 기능이 있다.

4.3 AHEAD의 전체적인 구성도

그림 6은 블럭 다이아그램으로 도시한 AHEAD의 전체적인 구성도이다. 본 시스템은 크게 클라스 브라우저, 스키마 정의기, 객체 데이타 편집기, 노드 및 링크 관리기, 질의 편집기, 브라우저, 스키마 관리기, 질의어 처리기, 저장 관리기로 구성되어 있다.

이중 질의어 처리기와 저장 관리기를 제외한 나머지 모듈들은 모두 상세히 구현되었으며, 질의어 처리기는 3.3절에서 언급한 바와 같이 질의어 최적화 과정을 제외한 나머지 부분을 프로토타입으로 구현하였다.

본 논문에서 구현한 저장 관리기는 다수의 사용자를 지원하기 위한 병행제어, 복구등의 기능을 가진 완전한 데이타베이스 저장 관리기의 기능을 가지고 있지 않다. 본 시스템에서는 Objective-C에서 지원하는 스트림 인터페이스(stream interface) 기능을 가진 객체 Archive 화일을 이용하여 객체들을 각 시스템 모듈들과는 무관하게 영속적으로 관리할 수 있는 기능만을 갖도록 구현하였다. 이와 같이 하므로서 저장된 객체들은 각 시스템 모듈들에서 공유할 수 있으며, 객체 단위로 입출력할 수 있는 가장 기본적인

기능을 지원하는 저장 관리기로서 사용할 수 있다.

저장 관리기는 인스탄스들을 저장하는 archInst와 원하는 인스탄스를 꺼내주는 unArchInst라는 메소드를 가지고 있다. 인스탄스를 생성하는 모든 응용들은 archInst 메소드를 이용하여 화일별로 인스탄스를 저장한다. 클라스의 인스탄스를 읽기 위해서는 unArchInst 메소드로 저장 관리기에 읽기 요청을 해야 하며 저장 관리기는 그 클라스의 인스탄스를 모두 넘겨주게 된다. 인스탄스가 변경되면 역시 archInst 메소드를 사용하여 재저장 하도록 구현하였다.

비록 여기서 구현된 저장 관리기는 기초적인 객체 저장의 기능만을 제공하지만, 객체 단위의 처리 기능이 제공되므로 일반적인 객체 지향 데이타베이스 시스템의 저장 관리기의 기본 형태라 볼 수 있으며, 추후 좀 더 완벽한 기능의 저장 관리기를 개발하여 대체할 계획이다.

AHEAD의 대화 관리기는 모든 윈도우를 관리하며, 메뉴의 제어를 담당하고, 사용자의 요구를 마우스 또는 키보드로 부터 입력받아 이를 사건(event)으로 처리한다. 즉, 모든 시스템 요소들의 제어를 담당한다.

클라스를 생성하고 생성한 클라스들의 계층구조를 쉽게 파악하기 위해서는 클라스를 브라우징해서 볼 수 있는 기능이 필요한데, 본 시스템에서는 다음 그림 7과 같은 클라스 브라우저를 제공한다. 클라스 브라우저는 응용의 계층 구조를 잘 표현할 수 있다. 예를 들면, 그림 7은 Student와 Professor 클라스는 Person의 하위 클라스로 정의되어 있음을 알 수 있다. 만일 새로운 클라스 Graduate를 생성하고자 한다면 Class 메뉴에서 create 기능을 선택하면 된다.

AHEAD에서는 하나의 응용을 생성하기 위해서는 먼저 그것의 스키마를 정의하도록 하고 있다. 특히 본 시스템에서는 각각의 응용들의 일반화 계층(generalization hierarchy)을 구성하고 있기 때문에, 하나의 응용을 생성하기 위해서는 먼저 어떤 클라스의 하위 클라스에 등록할 것인가를 결정하여야 한다. 임의의 한 클라스로 등록이 되면 그 클라스는 상위 클라스(super class)의 애트리뷰트 및 메소드를 상속받을 수 있다.

스키마 정의기는 논리 편집기와 배치 편집기로 나눌 수 있으며, 스키마 정의기에서 정의한 클라스와 애트리뷰트에 대한 정보는 스키마 관리기에서 관리하고 있다. 즉, 스키마 관리기에서는 ClassInfo 클라스와 Attr 클라스를 관리하며 스키마 정의에 대한 정

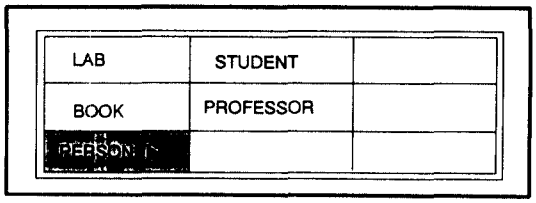


그림 7. 클라스 브라우저
Fig 7. Class browser

보를 제공하고 있다. 이에 대한 설명은 2.2절에 나타냈다.

논리 편집기는 크게 두 부분으로 구성된다. 그림 8의 왼쪽은 본 시스템이 제공하는 애트리뷰트의 타입을 나타내는 아이콘과 하나의 타입을 선택할 수 있는 라디오 버튼(radio button)으로 구성되며, 그림 8의 오른쪽은 애트리뷰트의 이름을 정의할 수 있는 부분과 라디오 버튼에 의해 선택된 타입을 정의할 수 있는 부분으로 구성되는데, 정의된 애트리뷰트의 종류가 많을 경우에는 수직 스크롤도 가능하다. 그림 8의 논리 편집기에서 오른쪽 부분의 굵은 선 위에 나타난 애트리뷰트들은 상위 클라스로부터 상속받은 애트리뷰트로서 자동적으로 화면에 나타나며, 사용자의 굵은 선 아래 부분에 자신이 원하는 애트리뷰트들을 정의할 수 있다.

논리 구조의 편집이 끝나면 배치 구조를 정의할 수 있는 배치 편집기가 나타나는데 하나의 클라스에 대해서 하나의 배치 구조만을 정의할 수 있다. 이 배치 편집기는 먼저 논리 편집기에서 정의되었던 애트리뷰트들의 디폴트(default) 배치가 나타나는데 사용자는 마우스 드래깅(mouse dragging)을 이용하여 그림 9와 같이 자신이 원하는 형태로 디폴트 배치를 재 배치할 수 있다.

객체 데이타 편집기에서는 텍스트 편집기, 이미지 편집기, 오디오 편집기, 그래픽 편집기, 그리고 정형 데이타 편집기를 지원한다. 텍스트 편집기는 APP KIT에서 제공되는 텍스트 패인을 이용하여 데이타를 편집할 수 있도록 한다. 이미지 편집기에서는 외부에서 원하는 Tiff 화일을 가져올 수 있도록 하며, 오디오 편집기는 응용 데이타에 표시된 오디오 아이콘에 대해 오디오 데이타를 정의할 수 있는 기능을 제공한다. 이 오디오 편집기는 Interface Builder에서 제공하는 Lip 기능을 이용해 구현하였는데, 여기에서는 사운드의 녹음(record), 플레이(play), 정지

(stop) 등의 기능을 제공한다. 실제 이러한 데이타들을 편집하여 저장할 때는 외부 BLOB(Binary Large OBject) 형태를 갖는 화일 이름으로 저장된다. 따라서 이 데이타들을 사용하기 위해서는 이 데이타를 조작(manipulate)하는 메소드에 원하는 화일 이름을 매개 변수(parameter)로 사용하여야 한다.

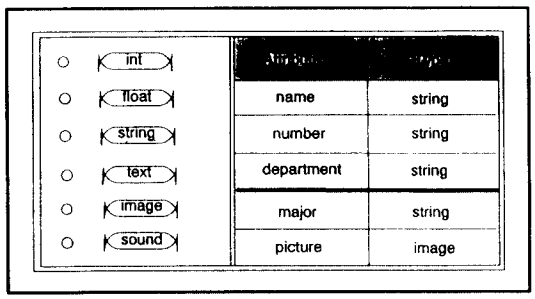


그림 8. 논리 편집기
Fig 8. Logical structure editor

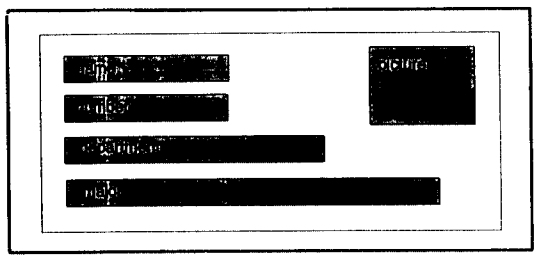


그림 9. 배치 편집기를 이용한 재배치
Fig 9. Repositioning by layout structure editor

V. 관련 연구 비교

본 장에서는 본 논문에서 설계, 구현한 AHEAD와 관련된 연구들을 비교 분석하였다. 크게 나누어 우선 기존의 하이퍼미디어 시스템들에 비교하여 AHEAD의 차이점을 분석하였고, 두번째는 기존에 연구되었던 객체 지향 개념을 기반으로 한 하이퍼미디어 또는 다매체 데이타베이스 시스템들과의 특징을 비교하여 AHEAD의 장점을 서술하였다.

5.1 기존의 하이퍼미디어 시스템과의 비교

기존에 구현된 많은 하이퍼미디어 시스템중에 대표적으로 손꼽을 수 있는 시스템으로는 브라운 대학의 Intermedia[6][12], XEROX사의 NoteCards[7], Tektronix의 Neptune[4]등을 들 수 있다[3]. 이들 시스템에서는 사용자를 위한 추가적인 도구를 제공함으로써 하이퍼미디어 모델의 단점 및 질의 기능을 보완하도록 하여 왔다. 이 절에서는 우선 각 시스템의 특징들을 살펴보고, AHEAD와의 차이점을 비교하도록 한다.

Intermedia에서는 Style, Context, Web, Box라는 도구들을 제공하고 있다. Intermedia에서는 Style을 이용하여 문서를 구성하며, 이들 문서들의 모임은 Context라는 도구를 사용하여 관리된다. 각각의 문서에는 여러 개의 노드가 존재할 수 있으며 각각의 노드에는 여러 개의 링크가 설정될 수 있다. 또한, 사용자가 노드에 설정된 링크들과 각 링크가 가리키는 목적지(destination)에 대한 정보를 쉽게 찾아낼 수 있도록 Web이라는 링크 지도(map)를 제공하며, Dialog Box라는 도구를 사용하여 노드들간에 설정된 링크의 정보를 더욱 자세히 알려주도록 하고 있다. 이러한 기능을 제공하기 위해서는 노드와 링크에 대한 정보를 정의할 수 있어야 하는데 Intermedia에서는 실제로 링크와 노드에 대한 애트리뷰트(attribute)를 기술할 수 있는 기능을 가지고 있다.

NoteCards 시스템은 노트 카드, 브라우저(browser), FileBox라는 도구들을 제공한다. 각 노트 카드는 편집이 가능하며, 각각의 노트 카드에는 노드가 설정될 수 있으며 여기에서 다른 노드 카드로 링크를 설정할 수 있는데, 이때 링크는 여러 가지 타입을 가질 수 있다. 브라우저는 노트 카드들의 네트워크를 구조적 다이아그램(structural diagram)으로 보여 주는 하나의 노트 카드이며 FileBox는 서로 링크된 노트 카드들의 집합이 어떻게 구성되어 있는지를 보여 주는 특별한 카드이다. 이것은 저장된 검색을 위한 계층적 분류 구조(hierarchical category structure)를 표현하기 위해 사용된다. NoteCards에서는 제한적이나마 탐색과 질의를 할 수 있는 기능이 제공되어 하이퍼네트워크에서 정보 검색에 도움을 주고 있다.

Neptune에서는 노드와 링크에 대한 저장 장치와 접근 방법(access mechanism)을 제공하면서 일반적인 하이퍼텍스트 모델을 표현하는 HAM 층과 그 위에 구성된 응용층(application layer)을 가진다. 이 응용층은 하이퍼텍스트를 자동적으로 다루고 변형시키는 프로그램들로 구성되어 있으며 이 응용층 위

에는 사용자 인터페이스(user interface)층이 존재하여 프리젠테이션을 위한 기능을 제공한다. 노드와 링크에 대한 시맨틱을 제공하기 위하여 Neptune은 이들에 대한 애트리뷰트를 정의할 수 있는 기능을 갖추어 노드가 가지는 속성과 노드 사이의 관계성을 기술할 수 있는 방법을 제공한다.

지금까지 살펴보았듯이 Intermedia, NoteCards 그리고 Neptune은 주로 노드에 설정된 링크 관계를 구조적으로 기술하거나 항해를 위한 링크의 지도를 보여줌으로써 하이퍼미디어 모델이 가지고 있는 단점을 해결하고자 노력하고 있다.

Intermedia와 Niptune은 노드와 링크에 대한 모델링을 통하여 풍부한 시맨틱을 제공하여 항해에 따른 어려움을 덜어주려고 노력하고 있으며 이 정보는 노드들을 구조적으로 구성하는데 유용하게 사용될 수 있다.

그러나 이들 시스템들에서는 응용에 나타나는 응용 데이타에 대한 복잡한 구조 및 관계성을 효과적으로 표현하는 능력이 매우 미흡하다. Neptune의 경우에는 정보 자체를 노드 단위로만 구성을 하여 이 점에 있어 더욱 어려움을 주며, Intermedia에서 Context는 단지 문서들의 집합만을 나타낼 뿐 문서의 계층 구조적 시맨틱을 반영하지 못하고 있다.

특히 NoteCards의 경우에는 노드와 링크에 대한 정보가 부족하며 노드 사이의 관계성은 링크의 선 형태가 갖는 시맨틱 외에는 별다른 시맨틱을 부여하지 못한다. 또한 Browse나 FileBox는 노드 사이의 구조를 표현함에 있어 단순히 이미 구성된 구조 정보만을 포함하고 있을 뿐 실제로는 노드들간의 구조에 대한 변화를 직접 반영하지 못한다.

또한 이들 모든 시스템에서의 매체 데이타는 단지 데이타만을 표현하는 수동적 수단이며, 이들 데이타를 다루기 위해서는 각 매체에 맞는 편집기가 제공되어야 가능하고 더우기 AHEAD에서와 같이 매체에 따른 다양한 연산을 제공하지 못한다.

한편 질의 능력에 있어서는 NoteCards의 경우에만 제한적인 탐색 및 질의 기능이 제공되며, 이 기능도 자체적인 탐색 기능에만 의존할 뿐 노드나 링크, 응용 데이타의 모델링된 정보를 효과적으로 사용하지 못한다는 문제점을 지니고 있다[5].

한가지 더 부연하여 강조하고자 하는 점은 AHEAD에서는 이들 모든 데이타의 모델링 및 질의 기능이 단순히 별도로 제공되는 도구에 의한 것이 아니라, 단일한 객체 지향 데이타 모델에 기반을 두어 통합적으로 지원된다는 것이 큰 장점이라 할 수 있으며, 이는 또한 시스템의 유연한 확장성을 제공하여 준다.

5.2 기존의 객체 지향 개념을 기반한 시스템들과의 비교

기존의 하이퍼미디어 또는 다매체 데이타베이스 시스템에 관한 연구중 본 논문의 AHEAD와 유사하게 객체 지향 개념을 이용하여 데이타 모델링 능력 또는 질의 능력의 향상을 꾀한 연구들이 있었는데, 대표적으로 [14][9][13][11][8] 등의 시스템을 들 수 있다.

본 절에서는 이들의 연구와 AHEAD의 차이점을 비교 분석하고자 한다. 비교의 기준을 삼기 위하여 본 논문의 연구 목적인 데이타 모델링 능력의 향상 측면과 질의 기능의 향상 측면으로 나누어 기존 연구를 비교한 결과를 표 1에 도시하였다. AHEAD에서는 데이타 모델링 능력에 있어서 하이퍼미디어 정보, 응용 데이타, 매체 정보의 모델링을 모두 지원하고

표 1. AHEAD와 기존 객체 지향 기반 연구와의 비교
Table 1. Comparison of AHEAD with previous researches based on object-oriented concept

지원 기능 / 기존 연구	데이타 모델링 능력의 향상			질의 기능의 향상			
	하이퍼미디어 정보의 모델링	응용데이타 모델링	매체 정보의 모델링	브라우징 및 항해 기능	객체 지향 대수 질의기능	하이퍼미디어 연산자 질의기능	매체 관련 연산자 질의기능
Woelk[14]	△	O	X	X	X	X	X
Masunage[9]	△	O	O	X	X	X	X
InterSect[13]	O	O	X	O	X	X	X
Harmony[11]	O	X	O	O	X	△	O
Ishikawa[8]	△	O	O	O	△	△	O

있으며, 질의 기능에 있어서는 브라우징 및 항해, 객체 지향 대수를 이용한 질의, 하이퍼미디어 연산자를 이용한 질의 및 매체 연산자를 이용한 질의를 공히 지원하고 있다. 표 1에서 'X'표는 이들 시스템중 지원을 하지 못하는 기능을 나타내고 있으며, '△'는 부분적으로 지원하고 있음을 나타내고 있다.

Woelk[14]는 다매체 데이타베이스에 객체 지향 데이타 모델링 개념을 도입한 거의 최초의 논문으로서 다매체를 이용한 응용에서 필요한 데이타 모델링 요건 및 기본 골격을 제시하고 있으나 이들 정보를 이용한 질의 기능은 제시되어 있지 않다. 이 논문에서는 다매체 응용 데이타를 클라스와 인스탄스를 위한 토큰 객체, 관계 객체, 속성, 메소드 객체, 본질 데이타 객체 등 몇가지 유형의 객체들로 구성하고 이들 객체들간의 관계를 설정하여 표현하고 있다. 또한 노드나 링크에 대한 자세한 정보의 모델링이 미흡하여, 각 매체 정보를 모델링할 수 있는 기능이 결여되어 있다.

Masunaga[9]의 논문은 객체 지향 개념에 기반을 두어, 응용 데이타 및 매체 정보의 모델링 방안을 제시한 비교적 초기의 논문으로 이 논문의 요점도 본 논문의 동기와 유사하게 하이퍼미디어 개념으로 응용 데이타 모델링이 어렵다는 점을 지적하고 있으며 특히 다양한 매체 모델링에 촛점을 두고 있다. 이 논문에서는 각 매체에 해당되는 개념 모델을 설정하고 다른 개념적 모델 내에 존재하는 객체들간의 연결이나 통합을 위해 다매체 링크를 제공하고 있다. 그러나, 이들 링크에 대한 추가 정보를 제공하지 못하고 있으며, 질의 기능 지원에 관한 언급이 없다.

InterSect[13]에서는 하이퍼미디어 모델의 장점과 객체 지향 데이타 모델의 장점을 모두 살리는 실험적 프로토타입을 구현하고 있으나, 여기에서도 역시 모델링에만 중점을 두고 있으며 이들 정보를 이용한 질의 기능은 제시되고 있지 않은 실정이다.

Harmony[11] 시스템은 객체 지향 개념의 강력한 모델링 능력을 이용하여 하이퍼미디어 정보 및 시간적이며 능동적인 매체 정보를 모델링하고, 메세지 메카니즘을 통해 이들의 연산을 지원하는 개방(open)화된 하이퍼미디어 시스템이다. 그러나 이 시스템에서의 데이타베이스의 의미는 단지 객체 단위의 저장소 구실만이며, 응용에서 나타나는 공통 특성을 갖는 객체들을 모델링하거나 집합 단위의 질의 능력을 지원하는 등의 다른 데이타베이스의 특성을 갖지 못한다. 질의 기능으로는 능동적 매체에 대한 연산 및 일반적인 항해와 브라우징을 제공하나 객체 지향 대수를 이용한 질의 기능은 지원하지 않으며 하이퍼미디어 연산자를 이용한 집합 단위의 질의 기능도 부분적으로 지원하고 있다.

Ishikawa[8]의 논문에서는 일반적인 객체 지향 지식 베이스 시스템으로부터 출발하여 이를 하이퍼미디어 엔진으로 사용할 수 있는 가능성을 제시하고 있다. 하지만 이 접근 방법에서는 본 논문에서와 같이 노드나 링크의 정보 및 이의 관련 연산을 통합하여 관리하지 못하고, 각 응용 객체 클라스의 애트리뷰트로서 링크를 표현하며, 하이퍼미디어 연산은 각 응용 클라스마다 메소드로 구현해 주어야 하는 어려움이 있다. 따라서 각 객체마다 노드와 링크의 시맨틱이 달라질 수 있어, 유연성은 있으나 복잡성이 높아질 수 있다. 또한 질의 기능을 향상시키기 위해 브라우징 및 항해, 매체 관련 연산자를 이용한 질의 기능을 제공하고 있으나 객체에 대한 질의는 객체 지향 대수를 기반으로 하고 있지 않고 단순히 경로 연산을 지원하며, 하이퍼미디어 연산자를 이용한 질의 기능은 부분적으로 제공하고 있다.

이와 같이 본 논문과 관련된 연구들을 종합적으로 살펴볼 때, AHEAD의 장점으로는 단일한 객체 지향 데이타 모델링 능력을 기반으로 하여 하이퍼미디어 모델의 비순차적 정보를 유지하며, 응용 데이타의 복잡한 구조를 추상화를 통해 효과적으로 표현하고, 매체 정보의 구조 및 매체 관련 연산을 모델링할 뿐만 아니라, 이들 정보를 이용한 집합 단위의 객체 검색 및 하이퍼미디어 정보 검색, 매체 관련 연산 지원등 다양한 형태의 질의를 지원할 수 있다는 것이다.

Ⅵ. 결 론

본 논문의 목적은 노드와 링크에 기반한 단순한 구조와 제한된 정보 검색 능력을 지닌 현재의 하이퍼미디어 시스템에 객체 지향 데이타베이스 기술에서 지원하는 강력한 데이타 모델링 능력과 다매체 데이타 처리 및 질의 기능을 결합하여 보다 향상된 하이퍼미디어 시스템인 AHEAD(A Hypermedia System with Enhanced Data Modeling and Querying Capability)를 설계, 구현하는데 있다.

기존의 하이퍼미디어 시스템에 비교하여 AHEAD가 가지는 장점은 다음과 같다.

첫째, AHEAD에서는 응용의 전체적인 구조와 하나의 응용 데이타에 대한 구조를 노드와 링크를 이용

하지 않고 직접 모델링을 통하여 시맨틱을 제공하고 있으므로 응용 데이타에서 실제로 필요한 노드와 링크를 정의하는 것 외에는 구조 정보 표현을 위해 수많은 노드와 링크를 설정할 필요가 없다.

둘째, 노드와 링크에 대한 모델링으로 연관된 노드 및 링크들 사이의 구조 정보를 효율적으로 표현할 수 있다. 이러한 장점은 기존 하이퍼미디어 모델에서 제공하지 못하였던 노드들의 계층적인 표현이나 구조적인 특정 조건을 만족하는 노드나 링크에 대한 검색을 가능하게 한다.

셋째, AHEAD에서의 질의 기능은 특정 조건을 만족하는 데이타 혹은 노드나 링크를 내용에 의한 검색이나 구조에 의한 검색을 지원한다. 질의 기능은 데이타 모델링을 통하여 얻는 부가적인 잇점으로 볼 수 있으며, 질의 기능의 지원은 연관된 자료의 탐색을 가장 효율적으로 지원하고, 특정 데이타의 검색 및 구조의 파악을 위해 행해질지도 모르는 수많은 항해를 피하게 한다.

넷째, AHEAD에서는 다매체 자료에 대한 모델링도 제공함으로, 다매체 자료에 대한 질의 및 처리 연산을 가능하게 한다.

향후 연구 계획으로는 저장 관리기 및 질의 처리기의 완전한 구현, 공간 및 시간적 동기화, 참조 링크와 주석 링크외에 링크 타입의 확장등을 고려하고 있으며, 타입 구성자의 확장에 대한 연구도 향후 연구 과제로 남아 있다. 그밖에, 버전(version) 관리 기능과 병행 제어, 복구등 다수 사용자 지원 기능 확장도 추후 연구 과제라 할 수 있다.

참 고 문 헌

1. Atkinson, M., et. al., "The Object-Oriented Database System Manifesto," Proc. of DOOD '89, pp.40-57, Dec., 1989.
2. Caudillo, R., and Mainguenaud, M., "Hypertext-like Multimedia Document Data Model," Int. Conference on Multimedia Information Systems, pp.221-241, 1991.
3. Conklin, J., "Hypertext : An Introduction and Survey," IEEE Computer, Vol. 2, No. 9, pp. 17-41, Sep., 1987.
4. Delisle, N. and Schwartz, M., "Neptune : A Hypertext System for CAD Application," Proc. of ACM SIGMOD Int'l Conf. on Management of Data, May, 1986, pp.132-143.
5. Fuller, M., et. al., "Querying in a Large Hyperbase," DEXA'91, pp.455-458, 1991.
6. Garret, K.E., et al., "Intermedia : Issues, Strategies, and Tactics in the Desing of a Hypermedia Document System," Proc. Conf. on Computer-supported Cooperative Work, MCC Software Technology Program, Austin, Texas, 1986.
7. Halasz, F.G., "Reflections on NoteCards : Seven Issues for the Next Generation of Hypermedia Systems," Communications of the ACM, Vol. 31, No 7, pp.836-852, Jul., 1988.
8. Ishkawa, H., et. al., "An Object-Oriented Knowledge Base Management System as a Next Generation of Hypermedia Engine," Proc. of Advanced Database System Symposium '89, pp.285-292, Dec., 1989.
9. Masunaga, Y., "An Object-Oriented Approach to Multimedia Database Organization and Management," International Symposium on Database Systems for Advanced Applications, pp.190-200, April, 1989.
10. Shaw, G.M. and Zdonik, S.B., "A Query Algebra for Object-Oriented Databases," Proc. of the Sixth International Conference on Data Engineering, pp.154-162, Feb., 1990.
11. Shimojo, S., et. al., "A New Hyperobject System Harmony : Its Design and Implementation," Proc. of the International Conference on Multimedia Information Systems '91, pp. 243-257, 1991.
12. Tankelovich, N., Haan, B.J., Meyrowitz, N. K., and Drucker, S.M., "Intermedia : The Concept and the Construction of a Seamless Information Environment," IEEE COMPUTER, Jan, 1988, pp.81-96.
13. Wang, B. and Hitchcock, B., "InterSect : A General Purpose Hypertext System Based on an Object Oriented Database," DEXA'91, pp. 459-464, 1991.
14. Woelk, D., Kim, W., and Luther, W., "An Object-Oriented Approach to Multimedia Da-

tabases," Proc. of the ACM SIGMOD'86, pp. 311-325, 1986.
15. 김정화, "객체 지향 및 하이퍼미디어 개념을 기반으로 한 다매체 데이타 모델링," 충남대 컴퓨터공학과, 석사학위 논문, 1992. 2.
16. 류은숙, "다매체 데이타베이스 언어의 요건 분석 및 설계," 충남대 컴퓨터공학과, 석사학위 논문, 1992. 2.
17. 송치봉, "객체 지향 개념을 기반으로 한 다매체 데이타베이스 질의어의 설계 및 구현," 충남대 컴퓨터공학과, 석사학위 논문, 1993. 2.

李 圭 哲(Kyu Chul Lee) 정회원
1984년 2월 : 서울대학교 컴퓨터공학과(졸업)
1986년 3월 : 서울대학교 컴퓨터공학과(석사)
1990년 8월 : 서울대학교 컴퓨터공학과(박사)
1989년 3월 ~ 현재 : 충남대학교 컴퓨터공학과 조교수
※주관심분야 : 객체-연역 데이타베이스 시스템, 멀티미디어 시스템, 소프트웨어 공학