# 시각적 객체 지향 데이타베이스 설계 지원 시스템에 관한 연구

류 시 국† · 강 현 석††


## 요 약

본 논문은 객체 지향 데이타베이스 설계의 전 과정을 지원해 주는 VODS의 설계 및 구현에 대해서 기술한다. VODS는 EOMT를 기반으로 데이타베이스 설계 과정을 수행한다. EOMT는 하이퍼미디어 객체를 다룰 수 있도록 하기 위해 OMT의 객체 모델에 정보항해 기능을 지원할 수 있도록 확장한 객체 지향 데이타베이스 설계 방법론이다. 데이타베이스 설계자는 VODS의 기본 화면에서 시작하여 마우스로 적절한 아이콘과 명령을 클릭하면서 설계의 전 과정을 수행할 수 있다.


# Visual Object-Oriented Database Design Support System(VODS)

Shi Kook Rhyoo† · Hyun Syug Kang††


## ABSTRACT

In this paper, the design and implementation of VODS is described. VODS is an object-oriented database design support system. VODS supports the whole database design steps according to the EOMT(extended OMT). EOMT as an object-oriented database design methodology is deviced by extending the object model of OMT (Object Modeling Technique) for supporting information navigations to manipulate hypermedia objects. Database designers can do the whole design steps by clicking mouse buttons on the appropriate icons or commands from the initial screen mode.


## 1. 서 론

복잡한 대규모 데이타베이스를 설계할 때 SQL과 같은 DBMS 언어를 직접 사용하는 것은 언어 자체가 너무 저수준이며 바람직한 소프트웨어 공학의 원리들을 해칠 수 있기 때문에 대개 바람직스럽지 못하다. 따라서 보다 높은 수준에서 체계적으로 저수준 언어로 사상하는 방법론이 필요하다. 이에 대한 대표적인 방법론이 ER(개체-관계성)[4] 모델이다.

그러나 CAD/CAM, CASE, OIS, AI, 멀티미디어 등 새로운 응용들이 많이 나타나면서 ER 방법론은 모델링 능력을 향상시켜야할 여지를 많이 가지게 되었다. 무엇보다 ER은 개체들의 구조를 세분화하고 필요에 따라 상세성을 추가할 수 있게 하는 일반화(generalization) 기능에 해당하는 성질을 갖고 있지 못하다. ER은 역시 관계성의 부구조화 기능도 부족

※이 논문은 1994년도 한국 학술 진흥 재단의 공모과제 연구비에 의하여 연구되었음.

† 정 회 원:경남전문대학 전자계산과 교수
†† 정 회 원:경상대학교 전자계산학과 교수
논문접수:1995년 9월 23일, 심사완료:1996년 1월 12일

하다. ER은 단지 연합(association)만을 제공하는 반면에 최근의 응용에서는 집성화(aggregation)도 함께 지원할 것을 요구한다. 그래서 최근까지 ER 방법을 확장하는 연구가 많이 이루어져 왔다[1]. 이 중의 하나가 객체 모델링 기법(OMT)의 객체 모델이다.

OMT[11]는 객체 지향 개념을 이용하여 소프트웨어를 체계적으로 개발할 수 있게 하는 개발 방법론이다. 이 기법은 객체 모델, 동적 모델, 기능 모델을 상호 관련시키면서 실세계를 모델링하여 시스템을 개발하는 방법론이다. 이 중 가장 핵심 모델이 객체 모델로서 주로 객체의 정적인 부분에 촛점을 맞춘다.

본 논문에서는 이러한 OMT의 객체 모델을 확장하여 EOMT를 만들었다. EOMT는 객체 지향 데이타베이스 설계의 전 과정을 지원하며, 특히 하이퍼미디어 응용을 위한 정보 항해 기능도 다룰 수 있다. 그리고 이 EOMT를 기반으로 시각적 객체 지향 데이타베이스 설계 도구인 VODS의 시제품을 개발하였다.

지금까지의 데이타베이스 설계 도구들은 주로 ER 방법론에 기초한 것이었다. 더구나 초기의 설계 도구들은 데이타베이스 설계의 전 과정을 지원해 주는 것이 아니라 ER도 편집기나 자료흐름도 편집기와 같이 개념적 설계 단계만을 지원해 주거나 논리적 설계 단계만을 지원해 주는 시스템들이었다[3]. 이들에 비해 MastER Plus, ERwin, Bachman Re-Engineering Product Set와 같은 상용 데이타베이스 설계 도구들은 요구 분석 단계에서 물리적 설계 단계까지 지원해 주는 시스템들이다[1]. 그러나 이 도구들도 모두 ER 모델을 기반으로 하고 있다.

이들에 반해 VODS는 객체 지향 개념에 기반을 둔 EOMT를 지원하는 데이타베이스 설계 도구이다. VODS는 다중 윈도우 화면에서 데이타베이스 설계자가 EOMT 다이아그램을 이용하여 시각적으로 데이타베이스를 설계해 나갈 수 있도록 하며, 그림그리기(drawing) 소프트웨어가 설계의 의미를 능동적으로 포착하면서 객체와 관계성을 추적할 수 있게 한다. 그래서 설계자는 이 VODS를 이용하여 객체 지향 데이타베이스 설계를 단계적으로 자연스럽게 진행해 나갈 수 있다[10].

본 논문의 구성은 다음과 같다. 2장에서는 우리가 사용할 객체 지향 개발 방법론의 기본이 된 객체 모델링 기법(OMT)에 대해 알아 본다. 3장에서는 OMT를 확장하여 고안한 객체 지향 데이타베이스 설계 방법론 EOMT를 설명한다. 특히, EOMT의 사용 방법에 대해 알아 본다. 4장에서는 EOMT를 지원하는 시각적 객체 지향 데이타베이스 설계 도구 VODS의 인터페이스를 기술한다. 5장에서는 결론 및 향후 연구 과제를 논의한다.

## 2. 객체 모델링 기법(OMT)

지난 십여년동안 소프트웨어 시스템을 공학적으로 개발하는 방법으로 구조적 기법[13]이 각광을 받아왔다. 이 방법은 시스템의 기능적 관점에 기저를 두고 시스템의 기능을 분할하는 방법을 사용한다. 그러나 이 방법은 소위 "소프트웨어 위기"를 해결해 주지는 못하고 있다.

최근 소프트웨어 개발 방법론으로 객체 지향(Object-Oriented) 기법들이 각광을 받고 있다[11, 5, 7]. 종래의 구조적 기법이 소프트웨어의 동적인 기능 측면에 촛점을 맞추고 있는데 비해, 객체 지향 개발 방법론들은 소프트웨어의 정적인 데이타 측면과 동적인 기능 측면을 모두 중시하고 시스템을 개발할 때 요구되는 여러가지 강력한 모델링 개념(일반화/특수화, 집단화 등)과 캡슐화 개념 등을 잘 지원하고 있기 때문이다.

이러한 객체 지향 개발 방법론들 중의 하나로 객체 모델링 기법(Object Modeling Technique, OMT[11])이 제안되었다. OMT는 GE(General Electric) 연구소에서 개발된 방법론으로 응용 시스템을 객체 모델, 동적 모델, 기능 모델이라는 세 가지 모델을 다이아그램을 이용하여 모델링하며 시스템의 분석, 설계(시스템 설계, 객체 설계), 구현의 전 과정에 일관된 객체 지향 개념을 이용한다. 그리고 이를 통해 실세계를 보다 정확하게 모델링하고 유지보수가 용이하며 재사용성이 높은 소프트웨어를 제작할 수 있도록 하여 소프트웨어의 생산성을 향상시키는 방법론이다.

이 방법은 요구 명세로 부터 시작된다. 이는 문제를 표현하는 문장 형식이다. 그런데 사용자나 개발자 또는 관리자의 요구를 표현한 문제 문장(problem statement)은 대개 불완전하고 규정에 맞지 않기 때문에 분석 단계에서 요구사항의 모호함(ambiguity)과 불일치성(inconsisitency)을 찾아내어 좀 더 명확하게 해주

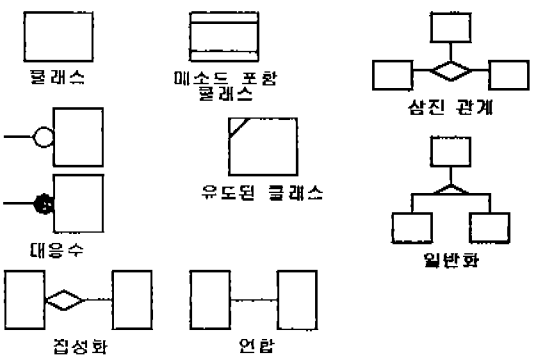
어야 한다. 이렇게 문제를 명확하게 노출시켜 실세계에 대한 모델을 만드는 과정이 시스템 분석 단계이다.

분석 단계에서는 세 개의 모델이 만들어지는데 객체 모델(Object Model)은 실세계를 나타내는 개체의 정적 구조를 나타내고, 동적 모델(Dynamic Model)은 일의 진행 순서를 나타내며, 기능 모델(Functional Model)은 자료의 변환(transformation)을 나타낸다.

이 단계에서의 객체 모델은 객체도를 이용하여 나타낸다. 실세계의 정적인 구조는 클래스들과 각 클래스의 애트리뷰트들(attributes), 연산들(operations), 그리고 그들 사이의 연합들(associations)로 나타내는데 객체 모델에서는 이들이 독립된 정보로 표현된다.

이러한 분석을 수행한 후 개발할 시스템을 개괄적으로 설계하는 것이 시스템 설계이다. 시스템 설계 동안에는 고수준의 시스템 구조가 선택된다(예, 일괄 변환 구조, 연속 변환 구조, 대화식 인터페이스 구조, 동적 시뮬레이션 구조, 실시간 시스템 구조, 트랜잭션 관리 구조 등).

시스템 설계를 마친 후 객체(상세)설계를 하게 된다. 객체 설계는 세 가지 분석 모델들을 더욱 세분화하여 구현에 대한 상세한 기초를 제공한다. 즉, 객체 설계에서는 분석 모델의 실세계 지향에서 시작하여 특정 구현을 위해 요구되는 컴퓨터 지향쪽으로 이동한다. 그러나 특정 언어나 데이타베이스 시스템을 고려함이 없이 시스템을 실현하는데 필요한 결정만을 수행한다.




(그림 1) OMT의 객체도 다이아그램들
(Fig. 1) Object diagrams of OMT

마지막으로 상세 설계에 따라 구현 및 유지보수 활동이 수행된다. 그리고 필요에 따라 뒷 단계에서 앞 단계로 언제나 피드백(feedback)이 일어날 수 있다.

이 기법의 핵심은 객체 모델이다. OMT에서 객체 모델은 시스템의 정적인 관점을 나타내는 모델로 객체도(Object Diagram)로써 표현된다. 즉, 객체도는 객체들(objects)과 클래스들(classes) 그리고 그들 사이의 관계성들(relationships)을 모델링하기 위한 정형적인 그래픽 표기법(formal graphic notation)으로 실세계의 추상화 모델링(abstract modelling)과 프로그램의 설계 작업을 위해 사용된다. 특히 이 객체도는 실세계의 정적 구조를 매우 쉽고 간결하게 나타낼 수 있는 다이아그램들로((그림 1) 참조) 데이타베이스 설계를 위한 모델링 도구로 사용할 수 있다[2].

## 3. 확장된 객체 모델링 기법(EOMT)을 이용한 데이타베이스 설계

이 장에서는 2장에서 기술한 OMT의 객체 모델을 확장하여 하이퍼미디어 응용의 데이타베이스 설계에 효과적으로 사용할 수 있는 EOMT를 제안한다. 우선 OMT의 객체도 모델링 능력에 정보항해를 위한 모델링 능력을 추가한다. 그리고 이를 데이타베이스 설계의 전 과정에 적용하는 절차를 고안한다.

### 3.1 OMT에 정보항해 기능의 확장

하이퍼미디어 응용에서는 클래스내에서 멀티미디어 자료와 같은 별도 관리가 필요한 자료들에 대한 검색 기능과 클래스들 사이의 관련된 자료 검색을 위한 항행 기능이 필수적이다[6, 12].

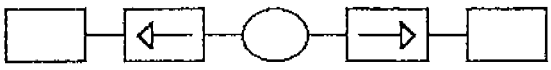
우리는 클래스내의 정보항해를 지원하기 위해 클래스에 포함된 서로 다른 미디어 자료들을 각각 단편들로 분리하고 하이퍼미디어 자료의 아이콘들로 이 자료들의 검색을 가능하게 한다. (그림 2)에 EOMT에

(그림 2) 단편 자료를 표시하기 위한 다이아그램
(Fig. 2) Slices diagram

서 단편들로 나누어진 자료들을 표시하는 다이아그램을 나타내었다. 이 다이아그램은 클래스 바깥쪽에 사각형을 추가하여 자료명을 적고 가위표를 이용하여 별도의 클래스로 표현된 단편들과 연결한다.

클래스들 사이의 정보항해를 위해서는 먼저 관련있는 클래스들을 그룹화하고 각 클래스내의 인스탄스들을 차례로 검색할 수 있는 기능과 클래스들 사이에 만들어진 색인을 이용한 항해가 가능하도록 해야 한다[8].

그룹화를 표시하기 위해서는 하나의 원을 사용해 그룹에 속하는 클래스들을 연결하고 각 클래스를 구성하는 인스탄스들을 차례로 탐색할 수 있는 기능은 그룹을 지정하는 원과 클래스 사이에 화살표가 포함된 사각형을 삽입한다((그림 3) 참조).



(그림 3) 그룹화와 인스탄스 항해 다이아그램
(Fig. 3) Grouping and guided tour diagram

클래스들 사이에 색인을 이용한 항해 기능을 지원하기 위해서는 이들 사이에 존재하는 연합의 대응수는 일대일 또는 일대다이어야만 한다. 따라서 다대다의 연합은 두개의 일대다 연합으로 분리하여 표현하여야 한다.

색인을 이용하여 다른 클래스의 관련 자료를 탐색하기 위한 다이아그램은 (그림 4(a))에서와 같이 테이블 형태를 포함하고 있는 사각형을 연합을 표시하는 선 사이에 나타낸다. 그리고 색인을 이용하여 찾은 후에 그 클래스내의 인스탄스들을 차례로 탐색하는 항해를 지원하기 위해서는 (그림 4(b))와 같은 다이아그램을 사용한다.

(a) (b)

(그림 4) 색인을 이용한 탐색 다이아그램과 정보항해 다이아그램
(Fig. 4) Indexed and indexed guided tour diagram

3.2 EOMT를 이용한 데이타베이스 설계 절차

EOMT를 이용한 데이타베이스 설계는 크게 4단계로 이루어 진다. 즉, 요구 분석(R), 개념적 설계(C), 논리적 설계(L), 물리적 구현(P) 단계들로서 RCLP 사이클이라고 명명한다. 이중 앞의 두 단계는 2장에서 기술한 OMT 객체 모델을 사용해서 시스템을 설계하는 것과 매우 유사하다. 그리고 나머지 두 단계는 관계형 데이타베이스 시스템을 위주로(반드시 관계형일 필요는 없으나 현재 가장 보편적으로 사용되고 있는 데이타베이스 시스템이다.) 실제 데이타베이스 구현에 까지 이르도록 한다. 이 방법은 기본적으로 문제 문장에서 물리적 스키마에 이르는 5수준 모델들을 4단계 사이클로 되먹임(feedback)을 통해 정제하며 설계해 나간다.

(1) 요구 분석 단계(R)

요구 분석은 문제 문장으로 기술된 실세계 모델(실세계 문제)에서 개괄적인 객체도로 기술된 분석 모델(데이타 분석도)을 만들어 낸다.

분석 모델을 만드는 첫번째 작업은 응용 영역으로부터 적절한 클래스들을 찾아내는 일이다. 클래스 객체들은 물리적 개체들(physical entities) 뿐만 아니라 개념들(concepts)을 포함해야 한다. 그런데 모든 클래스가 문제 문장안에 명확히 존재하는 것이 아니고, 어떤 클래스는 응용 영역이나 일반적인 지식(general knowledge)에 암시적으로 존재한다. 클래스를 찾기 위한 간단한 방법은 응용 영역에 관련된 요구 문장으로부터 명사를 추출하여 임시의 객체 클래스들을 만든 다음에 옳지 못한 클래스를 제거하면 정당한 객체 클래스가 생성된다.

둘 이상의 클래스들간에 나타나는 의존 관계나 한 클래스가 다른 클래스를 참조하는 관계를 연합이라 한다. 연합은 물리적 위치(next to, part of, contain in), 직접적인 행동(drives), 통신(talks to), 포함 관계(has, part of), 또는 조건의 만족(works for, married to, manages) 등을 포함한다.

하이퍼미디어 응용을 위해서는 클래스내의 멀티미디어 자료들을 여러개의 단편들로 나누어 관리한다. 각 단편들은 기본 자료내의 아이콘들을 이용해 검색할 수 있다. 여러가지 형태의 미디어 자료들을 기본 클래스와 관련지어 표현한다. 이의 표현은 대부분의

하이퍼미디어 응용에서 멀티미디어 자료를 관리하기 위해 거의 절대적으로 필요하다.

애트리뷰트는 이름, 무게, 속도, 색 등과 같은 각 클래스의 속성들이다. 두 클래스들 사이의 관계는 애트리뷰트로 나타내는 것이 아니고 연합으로 나타낸다. 유도된 애트리뷰트는 생략하거나 명확하게 분류한다.

상속을 사용하여 공통 구조를 공유하도록 클래스들을 구성할 수 있다. 상속은 여러 클래스들의 공통적인 면이 슈퍼클래스에 일반화되거나 슈퍼클래스의 정의를 상세하게 전문화함으로써 나타난다. 비슷한 애트리뷰트들, 연합들, 또는 연산들을 가진 클래스들을 일반화하는 과정에서 상속을 발견할 수 있다. 즉, 상향 일반화는 공통적인 속성들(common features)을 슈퍼클래스에 올려 정의함으로써 공유하는 것을 말한다. 하향 전문화(specialization)는 보다 상세하게 클래스를 세분화하여 정의하는 것을 말한다.

분석 모델링의 마지막 작업으로 시나리오(senario)를 이용하여 접근 경로를 검사한다. 객체도를 통한 접근 경로를 추적함으로써 의미있는 값들을 산출할 수 있다.

(2) 개념적 설계 단계(C)

개념적 설계는 개괄적인 객체도인 분석 모델(데이타 분석도)에서 상세한 객체도인 개념적 데이타 모델(고급 데이타베이스 설계도)을 만들어 낸다.

최초의 기본 설계 모델은 구현을 위한 골격으로서 요구 분석 단계에서 만들어진 분석 모델을 사용한다. 이 분석 모델은 시스템에 대한 논리적 정보를 상세히 표현하고 효율적인 정보 접근을 지원하기 위해 보다 세분화된다. 그리고 의미적으로 올바르기는 하지만 효율적이지 못한 분석 모델은 보다 효율적으로 구현될 수 있도록 최적화한다.

상속을 증진시키기 위해 클래스를 재정렬하고 연산(메소드)을 기술하고 조정한다. 이때 동일하지는 않지만 비슷한 공통 행위를 갖는 클래스들을 약간 변경한 후 결합하여 상위 클래스로 추상화한다. 즉, 매개인자의 수가 다르거나 더 특수한 연산, 또는 이름이 다른 비슷한 애트리뷰트들을 갖는 클래스들의 경우이다.

연합은 클래스들을 연결하는 아교 구실을 하는 것으로 연합을 구현하는 방법을 설계한다. 이를 위해 연합의 싸이클을 분석하고 각 연합을 구분된 클래스로 구현하거나 연합에 참여하고 있는 한두개의 클래스들에 연합에 관련된 애트리뷰트를 추가하여 구현한다.

클래스들 사이의 관련 자료 탐색을 위해 서로 탐색 가능한 클래스들을 그룹화한다. 정보항해를 위한 그룹화에 포함된 클래스들은 인스탄스들에 대한 탐색을 가능하게 구현할 수 있으며 클래스들간의 색인을 이용하여 관련된 클래스의 인스탄스 탐색도 가능하게 구현할 수 있다. 클래스들 사이의 정보항해를 위해서는 다대다 대응수를 갖는 연합은 두개의 일대다의 연합으로 분리하여 표현하여야 한다.

클래스의 표현은 비교적 똑바르다. 그러나 클래스 애트리뷰트의 정확한 표현을 결정해야 한다. 즉, 클래스들의 프리미티브 수준과 결합의 수준을 적절하게 결정해야 한다. 하나의 클래스를 여러 다른 수준의 프리미티브 클래스들로 표현할 수 있다.

(3) 논리적 설계 단계(L)

논리적 설계는 상세한 객체도인 개념적 데이타 모델(고급 데이타베이스 설계도)에서 DBMS에 독립인 이상적 관계형 데이타베이스인 논리적 데이타 모델(중급 데이타베이스 설계도)을 만들어 낸다.

첫 작업으로 고급 객체 구조를 DBMS에 독립된 일반형의 테이블로 사상한다. 자연스럽게 3차 정규형의 테이블이 되게 한다. 이를 위해 우선 객체는 직접 테이블로 사상한다. 이때 복합 객체는 분해한다. 일반화 관계성은 하나의 슈퍼클래스 테이블과 하나의 서브클래스 테이블로 사상한다. 그리고 같은 객체 식별자를 양 테이블에 모두 나타낸다. 다대다 집성화 관계성은 하나의 독립된 테이블로 사상한다. 일대일과 일대다 집성화 관계성은 내용에 따라 독립된 테이블로 사상하거나 관련 객체 테이블에 통합한다. 연합 관계성은 독립된 테이블로 사상한다.

각 객체 및 관계성에 유일한 객체 식별자를 후보키로 삼는다. 이는 각 객체를 참조하는 메카니즘으로 사용된다.

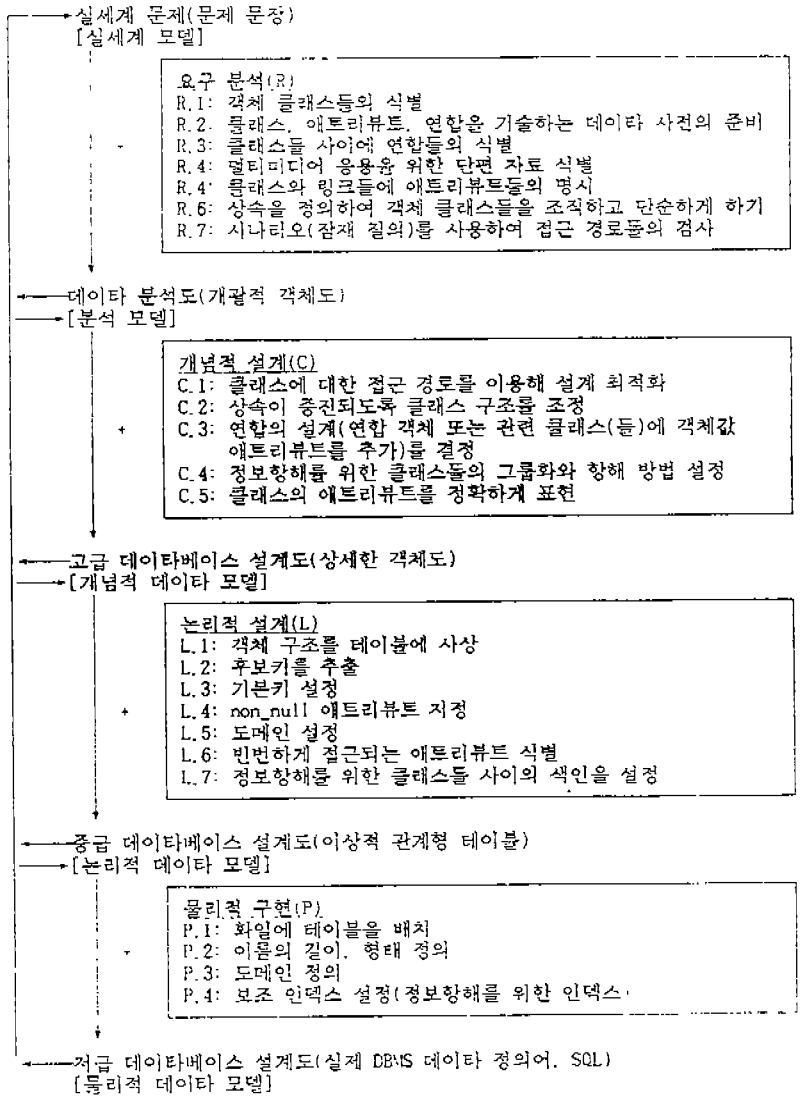
기본키는 튜플을 가장 빨리 접근할 수 있는 메카니즘이다. 후보키들 중의 하나를 기본 키로 설정한다. 기본키에 참여하는 필드는 null이 허용되지 않는다. 관계성 테이블의 경우 여러 식별자들이 기본키에 참

여하게 되지만 DBMS에 따라서는 하나의 애트리뷰트만 기본키로 허용한다. 이 경우에는 적절한 조정이 필요하다. 후보키에 참여하는 애트리뷰트들은 non-null로 지정한다.

각 애트리뷰트가 가질 수 있는 값의 범위를 설정한다. 이는 데이타의 일양성(consistency)을 위해 반드시 필요하다.

객체에서 빈번하게 접근하게 되는 애트리뷰트들의 그룹을 지정한다. 이는 앞으로 인덱싱(indexing)이나 해싱(hashing)의 기본적인 대상이 된다.

클래스들 사이의 정보항해를 위한 색인을 설정한다. 이 색인은 일대다 대응수를 갖는 연합에 관련된 클래스들의 기본키값들로 구성된다.




(그림 5) EOMT를 이용한 DB 설계 방법론(5단계 모델에 기반한 RCLP 사이클)
(Fig. 5) DB design methodology using EOMT

(4) 물리적 구현 단계(P)

물리적 구현은 이상적 관계형 데이타베이스인 논리적 데이타 도델(중급 데이타베이스 설계도)에서 특정 DBMS에 종속된 물리적 데이타 모델(데이타베이스 정의어)을 만들어 낸다.

첫 작업으로 각 테이블에 대해 화일을 지정한다. 이 지정은 특정 DBMS의 데이타 정의어로 이루어 진다. 사용가능한 화일의 형식은 해당 DBMS에 따라 달라진다.

각 테이블의 애트리뷰트에 대해 이름의 형식, 크기를 결정한다. 이 역시 해당 DBMS가 사용하는 화일 시스템에 따라 달라진다.

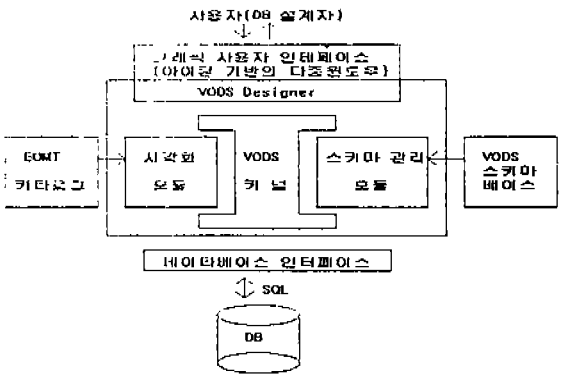
각 애트리뷰트가 가질 수 있는 값의 범위를 가능한 한 상세히 명세한다. 그리고 디폴트 값, 도메인의 마스크(mask) 등을 정의한다.

보조 인덱스는 어떤 애트리뷰트를 통해 빠르게 튜플을 접근하고 후보키의 유일성을 강화할 때 사용된다. DBMS에 따라서는 보조 인덱스가 여러 애트리뷰트들의 집단으로 정의될 수 없다. 이 경우 기본키와 보조 인덱스를 바꾸어야 할 지도 모른다.

(그림 5)에 RCLP 싸이클의 전체적인 설계 단계들을 요약하여 나타내었다.

## 4. 시각적 객체 지향 데이타베이스 설계 도구

이 장에서는 3장에서 설명한 데이타베이스 설계 방법론 EOMT를 지원하는 시각적 객체 지향 데이타베이스 설계 도구(VODS)를 기술한다. VODS의 개괄적인 구조는 (그림 6)과 같다.




(그림 6) VODS의 개괄적인 구조
(Fig. 6) Structure of VODS

VODS는 VODS 커널, 사용자 인터페이스 모듈, 시각화 모듈, 스키마 관리 모듈, 데이타베이스 인터페이스 모듈로 구성된다. VODS 커널의 제어하에 시각화 모듈과 스키마 관리 모듈이 각각 EOMT 카탈로그와 VODS 스키마베이스의 지원을 받아 데이타베이스 설계자의 설계 사항을 관리하며, 사용자는 이들을 사용자 인터페이스를 사용하여 이용하게 된다. 그리고 설계 활동에서 발생하는 모든 설계 정보와 카탈로그, 스키마베이스는 데이타베이스 인터페이스를 통해 DB로 관리된다. 이 장에서는 주로 Visual Basic으로 Window 3.1 환경에서 프로토타입으로 구현된 사용자 인터페이스를 통해 VODS의 특성을 알아 보기로 한다.

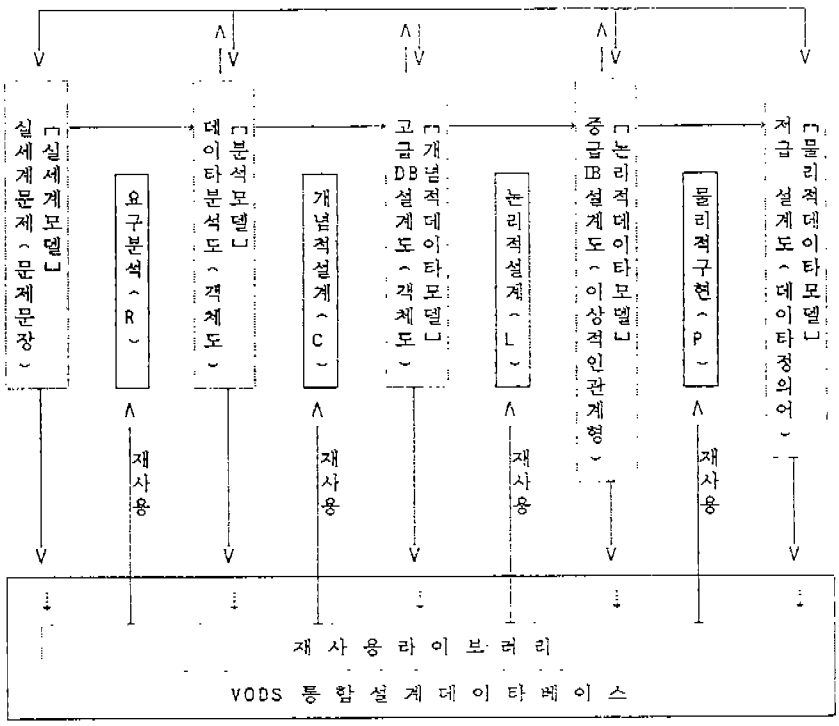
### 4.1 VODS를 사용한 EOMT 데이타베이스 설계 기법

EOMT에서는 객체 지향 데이타베이스를 VODS의 재사용 라이브러리의 지원을 받아 스키마재사용에 기초한 5수준 모델들을 4단계 RCLP 사이클로 되먹임(feedback)을 통해 정제하여 설계해 나간다. 즉, 반자동(semi-automatic)으로 각 데이타베이스 설계 단계에서 설계자의 결정을 받아 VODS가 모델들 사이를 사상하고 오류를 점검해 준다. 이러한 과정에서 생긴 각 수준의 데이타베이스 모델들은 통합 데이타베이스로 관리되고 다시 다른 데이타베이스(또는 되먹임 과정에서 자신의 데이타베이스) 설계에 사용되도록 재사용 추출 과정(→로 표시)을 거쳐 재사용 라이브러리로 관리된다. 이의 개괄적인 과정이(그림 7)에 나와 있다.

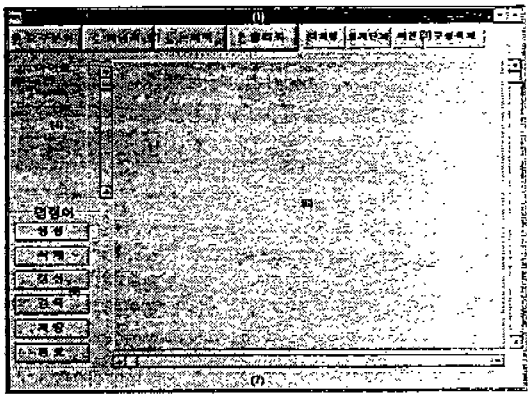
### 4.2 VODS의 인터페이스 구조

VODS 인터페이스의 기본 화면 형식은 (그림 8)과 같다.

위상 표시 영역으로 현재 설계중인 객체명(클러스터명), 설계 단계, 설계 중인 객체의 버전 번호, 설계 중인 객체에서 현재 다루고 있는 구성 객체의 이름들을 나타내 주는 영역이다. (4)는 원시 객체 아이콘 메뉴 영역으로 각 모델의 설계에서 사용되는 아이콘 메뉴들로서 각 단계마다 달리 나타난다. (5)는 객체 명령어 메뉴 영역으로 현재 화면(작업)에서 사용할 수 있는 명령어를 나열한다. 생성, 삭제, 갱신, 검색, 저




(그림 7) VODS의 지원을 받는 EOMT 사용 환경
(Fig. 7) EOMT usage environment in VODS




(그림 8) VODS 사용자 인터페이스의 기본 화면
(Fig. 8) Basic screen of VODS user interface

장, 종료 명령이 있으며 모든 경우에 있어서 아이콘과 이 메뉴 중의 명령이 연속적으로 클릭되면 메시지 영역(7)에 처리를 위한 과정을 화면에 표시해 주게 된다. 아이콘과 생성 명령이 클릭되면 클래스, 연합, 인덱스등을 만들기 위한 그래픽 기호들이 화면에 나타난다. 아이콘과 검색, 수정, 삭제 명령들 중에 하나가 클릭되면 화일 선택상자가 나타나서 검색이나 수정 또는 삭제할 클래스를 지정할 수 있게 한다. 저장 명령의 경우 클래스나 연합들을 화일로 저장할 수 있게 한다. (6)은 편집 화면으로 데이타베이스 설계자가 해당 단계의 아이콘과 명령어 메뉴를 이용하여 설계 객체들을 편집하는 영역이다. (7)은 메시지 영역으로 화면 상의 아이콘이나 명령어에 커서가 옮겨지면 해당 항목에 대한 도움말을 나타내거나 오류 사항에 대한 메시지를 출력한다.

(1) 요구 분석 단계와 아이콘과 사용 방법

이 단계의 아이콘들로는 클래스 아이콘, 연합 아이콘, 일반화 아이콘, 집성화 아이콘, 단편 아이콘들이 있다.

편집 영역에서의 기본 동작은 원하는 아이콘과 원

하는 명령의 선정으로 이루어 진다. 이는 모두 마우스의 클릭으로 이루어진다. 예를 들어 하나의 클라스를 생성하기 위해서는 클래스 아이콘을 클릭한 후 생성 명령을 클릭해야하며, 이 경우 편집 화면의 오른쪽 윗 부분에 상자가 나타나고 상자 안에 클래스명을 적고 상자를 원하는 위치로 옮기기 위해 마우스를 그 클래스에 위치시켜 누른 상태로 원하는 위치에 이동시켜 떼면 된다.

클래스들 사이를 관련시키기 위해서는 연합 아이콘과 생성 명령을 클릭한 후 편집 화면에 있는 해당 클래스들을 클릭하면 두 클래스를 잇는 선이 나타난다. 이 선을 클릭한 후 마우스가 위치한 곳에 연합명을 적을 수 있다.

일반화 연합의 경우 편집 화면에서 첫번째 클릭된 클래스가 일반화 계층에서 상위의 클래스가 된다. 생성된 일반화 계층에 하위의 클래스를 추가하려할 때는 일반화 아이콘을 클릭한 후 수정 명령을 클릭하고 상위 클래스와 추가할 하위 클래스를 차례로 클릭한다.

집성화 연합의 경우도 비슷하게 편집 화면의 첫번째 클릭된 클래스와 다음에 클릭된 클래스 사이에 집성화 아이콘으로 연결된다.

하이퍼미디어 응용을 위한 단편들을 나타내려하는 경우에는 이를 위한 아이콘과 생성 명령을 클릭하면 단편 자료명을 적을 수 있는 상자가 나타나고 여기에 단편 자료명을 표시한 후 상자를 클래스에 붙도록 위치를 조정하고 해당 클래스를 한번 클릭하여 그 클래스에 속하는 단편 자료명인 것을 표시한다. 다시 단편 자료명이 표시된 상자를 클릭하고 이 속성에 관련된 클래스를 클릭하면 이들 사이에 가위표가 나타나게 된다.

요구 분석 단계를 수행하는 화면이 (그림 9)이며 굵은 화살표는 현재 다루어지고 있는 Product 객체를 가리키고 있다.

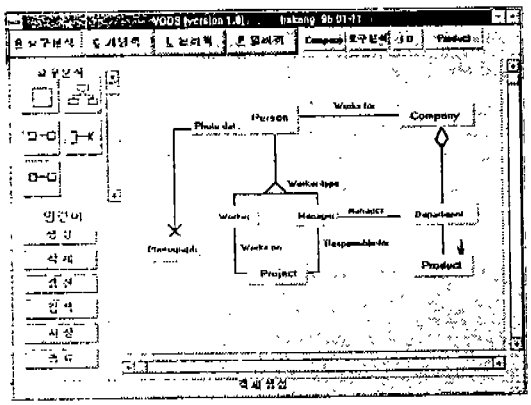
(2) 개념적 설계 단계의 아이콘과 사용 방법

이 단계의 아이콘들은 메소드 포함 클래스 아이콘, 유도된 클래스 아이콘, 대응수 제약 아이콘, 그룹화 아이콘, 인스탄스 검색 아이콘, 정보항해 아이콘 등이 있다.

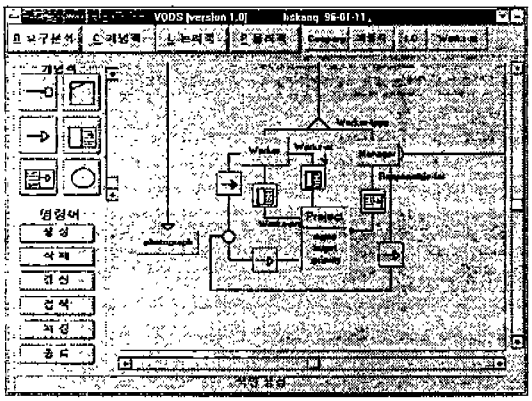
메소드를 포함한 클래스 아이콘과 수정 명령이 클릭되면 메소드를 추가할 클래스를 지정하기 위한 화일 선택 상자가 나타나고 지정한 클래스에 메소드를 추가할 수 있는 상자가 나타난다.

유도된 클래스의 경우에는 별도의 아이콘을 두어 이 아이콘이 클릭되면 유도된 값을 구하는 식을 적을 수 있는 상자가 나타나며 그 내용에 따라 클래스를 생성한다.

대응수 제약 아이콘의 경우는 연합, 집성화 등을 나타내는 선의 끝 부분에 작은 원을 표시하게 되는데 원의 내부가 채워진 경우는 1 이상의 대응수를 갖는 경우이며 채워지지 않은 경우는 0 이상의 대응수를




(그림 9) 요구분석 단계를 수행하는 화면의 예
(Fig. 9) VODS helping the designer to do requirement analysis



(그림 10) 개념적 설계 단계를 수행하는 화면의 예
(Fig. 10) VODS helping the designer to do conceptual design

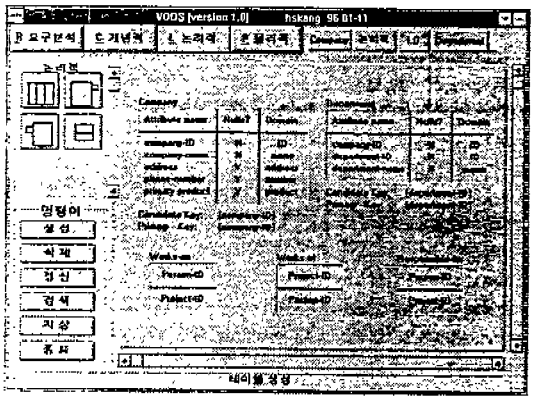
갖는 것을 표시한다. 하나의 대응수만을 가질 경우는 작은 원을 클릭한 후 마우스를 적당한 위치에 이동시켜 1로 대응수를 적어주며, n에서 m사이의 대응수를 가질 경우에는 n-m과 같이 표시하며, n이상의 대응수를 가질 경우는 n+와 같이 표시한다.

인스탄스 검색과 클래스 사이의 색인을 이용하는 정보 검색을 위해서는 클래스들을 그룹화해야 한다. 그룹화 아이콘과 생성 명령을 클릭한 후 해당 클래스들을 차례로 클릭하면 클래스들이 그룹화를 나타내는 원에 화살표를 포함하고 있는 사각형으로 연결된다. 인스탄스 검색을 위해서는 인스탄스 검색 아이콘과 생성 명령을 클릭하고 해당 클래스를 클릭하면 그룹화를 표시하는 원과 클래스 사이에 두개의 작은 사각형이 연결된 모양을 포함하는 사각형이 나타난다. 정보항해 아이콘과 생성 명령을 클릭하고 해당 클래스들을 클릭하면 이 클래스들 사이의 연합이 다대다 대응수를 갖는 경우 두개의 연합으로 갈라지게 되며 이 때 연합명을 표시하면 연결된 두개의 작은 사각형 모양과 화살표를 포함하는 사각형이 나타난다.

요구 분석 단계를 거쳐 개념적 설계 단계를 수행하는 화면이 (그림 10)이며 정보항해를 위한 색인 생성 화면이다.

(3) 논리적 설계 단계의 아이콘과 사용 방법

이 단계의 아이콘들은 테이블 아이콘, 기본키 아이콘, 후보키 아이콘, 항해색인 아이콘들이 있다.



(그림 11) 논리적 설계 단계를 수행하는 화면의 예
(Fig. 11) VODS helping the desiner to do logical design

테이블 아이콘과 생성 명령이 클릭되면 화일 선택 상자가 나타나며 지정된 클래스나 연합의 구성을 위한 테이블이 나타난다. 이 테이블은 애트리뷰트 수만큼의 행을 가지며 3열을 갖는데, 첫번째 열은 애트리뷰트명들이 적혀진 상태다. null값을 포함할 수 있는지의 여부를 결정하는 두번째 열과 도메인을 표시하는 세번째 열의 내용은 사용자가 적어준다.

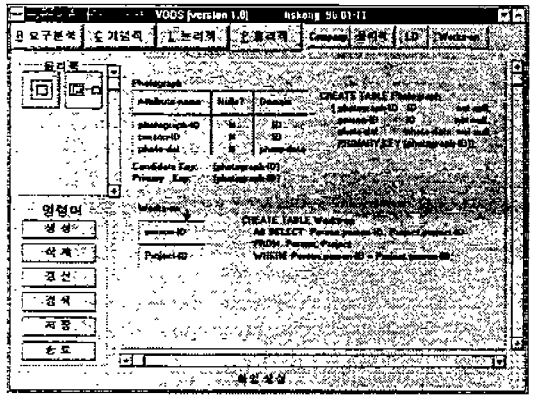
기본키 아이콘이나 후보키 아이콘이 클릭되면 테이블의 아래에 기본키 또는 후보키를 적어줄 수 있게 된다.

항해색인 아이콘과 생성 명령이 클릭되면 화일 선택 상자가 나타나며 하나의 클래스를 선택한 후 적용 버튼을 클릭하고 두번째 클래스를 선택하고 예 버튼을 클릭하면 각 클래스의 기본키 값이 적힌 상자가 나타난다.

논리적 설계 단계를 수행하는 화면이 (그림 11)이다.

(4) 물리적 설계 단계의 아이콘과 사용 방법

이 단계의 아이콘들은 화일 아이콘과 인덱스 아이콘이 있다.



(그림 12) 물리적 데이타베이스 설계 화면의 예
(Fig. 12) VODS helping the designer to do physical design

화일 아이콘과 생성 명령이 클릭되면 화일 선택 상자가 나타나며 화일 집중에 대한 정보를 마치지 않은 경우에는 적용 버튼을 클릭하며 화일에 집중될 클래스의 선택이 끝난 후에 예 버튼을 클릭한다.

인덱스 생성 아이콘은 인덱스를 구성할 에트리뷰트를 저장할 수 있도록 한다. 물리적 설계 단계의 결과는 특정 DBMS의 데이타 정의어로 출력된다. 우리는 이를 데이타베이스 표준 언어인 SQL로 출력한다. 물리적 데이타베이스 설계를 마친 후의 화면이 (그림 12)이며 정보항해를 위한 색인에 대한 화일을 생성한 화면이다.

## 5. 결론 및 향후 과제

OMT는 소프트웨어 시스템 개발에서 현재 가장 널리 사용되는 객체 지향 개발 방법론 중의 하나이다. 본 논문은 CAD/CAM, OSI, 멀티미디어등의 새로운 데이타베이스 응용들을 다루기위해 이 OMT를 확장한 EOMT 기법의 데이타베이스 설계 방법론을 제안하였다. 그리고 EOMT를 지원하는 시각적 객체 지향 데이타베이스 설계 도구 VODS를 설계/구현하였다. 이러한 EOMT 기반의 데이타베이스 설계 접근의 장점은 다음과 같다.

(1) 직감적이고 이해하기 용이하다. 비전문가라도 쉽게 사용할 수 있다.
(2) 표현력이 강하다. 데이타 모델링을 위해 풍부한 구조를 갖고 있다.
(3) 확장 가능하다. 다양한 데이타 모델에 적용하며 여러 DBMS에 이식 가능하다.
(4) 유용한 수준의 추상화를 제공한다. 실세계 문제를 잘 일치시키며 관계 DBMS에 자연스럽게 사상한다.
(5) 하이퍼텍스트(Hypertext)의 링크(link)를 모델링할 수 있다. 새로운 데이타베이스 응용에 사용할 수 있다.

현재 VODS의 개괄적인 설계가 완료되었으며 사용자 인터페이스에 대해서만 프로토타입으로 Visual Basic을 이용하여 Window 3.1에서 구현하였다. 구현할 때 설계의 각 단계에서 중간 결과를 보관했다가 다시 로드시켰을 경우 화면에서 그림과 자료 화일 간의 사상을 해결하기 위해 그림의 각 객체에 대한 위치 정보와 객체 사이의 관계에 대한 정보를 별도의 화일에 보관하였으며 다시 로드시킬 때 마다 이 정보를 이용하여 그림을 그렸다. 앞으로 EOMT를 지원하는 완전한 VODS의 설계 및 구현이 필요하며 워크스테이션급에서도 사용할 수 있도록 Motif를 이용하여 X-window에서 구현하는 작업도 필요하다. 그리고 HyTime [9] 등의 지원을 위해 시간을 모델링하는 기능도 포함시켜야 한다.

## 참 고 문 헌

[1] J. C. Batin, S. Ceri, and S. Navathe, Conceptual Database Design:An Entity-Relationship Approch, Benjamin/Cummings Publishing Company, pp. 411-454, 1992.

[2] M. Blaha et al., "Relational Database Design Using An Object-Oriented Methodology," Communications of the ACM, Vol. 31, No. 8, pp. 414-427, April 1988.

[3] R. Braegger, A. Dudler, J. Rebsamen, and C. Zehnden, "Gambit: An Interactive Database Design Tool for Data Structures, Integrity Constraints, and Transactions," IEEE Transactions on Software Engineering, SE-11, No. 7, pp. 574-583, July 1985.

[4] P. Chen, "The Entity-Relationship model: Toword a Unified View of Data," ACM TODS, Vol. 1, No 1, Mar. 1976.

[5] P. Coad and E. Yourdon, Object-Oriented Analysis, Englewood Cliffs, New Jersey, Yourdon Press, 1990.

[6] J. Conklin. "Hypertext: An Introduction and Survey." IEEE Computer, pp. 17-41, September 1987.

[7] B. Henderson-Sellers and J. Edwords, "The Object-Oriented Systems Life Cycle," Communications of the ACM, Vol. 33, No. 9, pp. 142-159, Sept. 1990.

[8] T. Isakowitz, A. Stohr, and P. Balasubramanian, "A Methodology for Structured Hypermedia Design," Communications of the ACM, Vol. 38, No. 8, pp. 34-44, Aug. 1995.

[9] S. Newcomb et al., "HyTime: The Hypermedia/

Time-Based Document Structuring Language," Communications of the ACM, Vol. 34, No. 11, pp. 67-83, Nov. 1991.

[10] T. Rogers and R. Cattell, "Entity-Relationship Database User Interfaces," IEEE Proc. of Data Engineering, pp. 44-52, 1988.

[11] J. Rumbaugh et al., Object-Oriented Modeling and Design, Prentice Hall, 1991.

[12] R. Vetter, C. Spell, and C. Ward, "Mosaic and the World-Wide Web," IEEE Computer, pp. 49-57, Oct. 1994.

[13] E. Yourdon and L. Constantine, Structured Design, Englewood Cliffs, New Jersey, Yourdon Press, 1979.

**류 시 국**

1980년 경북대학교 전자공학과 졸업(학사)

1989년 영남대학교 대학원 전자공학과(공학석사)

1996년 경상대학교 대학원 전자계산학과 박사과정 재학중

1980년~1996년 경남전문대학 전자계산과 부교수

관심분야:데이타베이스(특히, 객체지향 데이타베이스)

**강 현 석**

1981년 2월 동국대학교 전자계산학과 졸업

1983년 2월 서울대학교 계산통계학과 이학석사(전산학)

1989년 서울대학교 전자계산학과 이학박사(전산학)

1981년~1984년 한국전자통신연구소 연구원

1984년~1992년 전북대학교 전자계산학과 부교수

1992년~현재 경상대학교 컴퓨터과학과 부교수

관심분야:객체 지향 데이터베이스, 컴퓨터 그래픽스, 멀티미디어