IRDS C CLI의 설계 및 구현

박 중 기[†] 진 성 일[††] 이 헌 기[†††]

요 약

통합된 CASE(Computer-Aided Software Engineering) 도구의 개발 환경에서 소프트웨어 시스템의 재사용성을 높이고 정보를 공유하기 위해서는 정보저장소를 중심으로 하는 통합된 소프트웨어 환경을 개발하는 것이 필요하다. 본 논문에서는 통합 CASE 도구의 정보저장소로서 사용될 수 있도록 ANSI IRDS 표준에 따라서 이미 구현한 IRDS 서비스 인터페이스 프로토타입 시스템과 CASE 도구들 간에 정보저장소 인터페이스를 설계하고 구현한 결과를 논한다. 정보저장소 인터페이스 방법은 주 언어(host language)를 C로 하여 C 프로그램에서 IRDS 라이브러리 함수들을 호출하여 정보저장소를 접근하여 원하는 정보를 구조체로 받아 올 수 있도록 지원하는 CLI(Call Level Interface) 방법을 이용하여 구현되었다.

The Design and Implementation of C CLI for Information Resource Dictionary System

Joong Ki Park,[†] Seong Il Jin[††] and Hun Ki Lee[†††]

ABSTRACT

For the purpose of the sharing and reusability of software systems in an integrated CASE environment, it is necessary to develope the repository-based integrated software environment. In this paper, we designed and implemented the repository interface between CASE tools and IRDS service interface. We have already implemented the IRDS service interface system according to ANSI IRDS standard in order to develop a repository of an integrated CASE environment. We implemented the repository interface using the CLI(Call Level Interface) method that is composed of callable run-time libraries.

1. 서 론

소프트웨어 시스템의 생산성과 신뢰성 문제를 해결하기 위하여 소프트웨어 생명주기 전반에 걸쳐 CASE 도구의 활용이 증가하고 있다. 그러나 여러 종류의 자동화된 도구들이 하나의 개발 방법론 아래에서 일관된 사용 방법을 가지지 않고 각각의 도구들을 위한 정보저장소가 등장하게 되었다. 이러한 정보저장소는 같은 기능을 가지는 CASE 도구들이 서로 다른 정보저장 형태를 가짐으로서 상호간에 저장 정보를 공유하지 못하는 문제점을 갖게 되었고 각각의 CASE 도구에 따라 정의된 개념 모델에 준하여 메타 정보를 저장함으로써 각 도구들 간의 상호 연관성을 유지할 수 없게 되었다[2, 3, 9, 12].

CASE 도구에서 생성되고 관리되는 많은 정보들을 통합하여 저장하고 일관성 있게 관리하기 위하여 데이터베이스 관리 시스템(Database Management System: DBMS)을 이용하여 정보저장소의 기능을 할 수 있도록 하고 있다. 그러나 현재 대부분의 DBMS는 메타 정보를 정의하고 관리하는데 있어 그 기능이 매우 미약하며 자료의 사용이나 유지를 기술하고 제어하는 스키마 정보(Schema Information)의 제어나 전 조직체의 정보 관리를 위하여 필요로 하는 정보 자

† 정 회 원 : 시스템공학연구소 S/W공학연구부 연구원
†† 정 회 원 : 충남대학교 컴퓨터과학과 학과장
††† 정 회 원 : 시스템공학연구소 연구원
논문접수 : 1995년 3월 22일, 심사완료 : 1995년 6월 23일.

원(Information Resource)의 관리 능력이 부족하다[2, 10]. 따라서 DBMS을 CASE 도구의 정보저장소로 이용하기에는 많은 제약이 따른다.

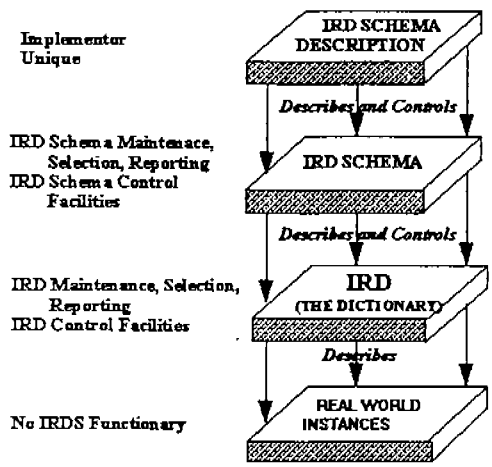
그러므로 이러한 정보 자원을 보다 완벽하고 효율적으로 관리할 수 있는 도구가 필요하다. 이러한 정보 자원을 효율적으로 통합, 저장, 관리하며 공유할 수 있는 기능을 제공할 수 있는 시스템이 바로 정보 자원 사전 시스템(Information Resource Dictionary System: IRDS)이다[1, 4, 6, 7, 8, 13, 17, 18]. IRDS는 미국 국립 표준국(American National Standards Institute: ANSI)과 국제 표준화 기구(International Organization for Standardization: ISO)에서 CASE 도구의 표준 정보저장소로 채택되었다[9]. IRDS는 정보 자원과 메타 정보를 관리하는 소프트에어로서 정보 환경을 표현할 수 있는 모델링 방법을 제공하며, CASE 도구를 위한 메타 정보를 정보 자원 사전(Information Resource Dictionary: IRD)이라고 불리는 공유 가능한 저장소(repository)에 저장하고 접근하고 관리하며 정의한다.

이러한 IRDS가 소프트웨어 생명 주기(software life cycle)를 지원하는 통합 CASE 도구의 정보저장소로 사용되기 위해서는 IRDS와 CASE 도구들 간에 정보저장소 인터페이스가 필요하다. 이러한 인터페이스를 제공하는 방법은 IRDS 명령어를 주 언어에 삽입(embedded)하는 방법과 IRDS 명령어를 처리할 수 있는 라이브러리 함수를 이용하는 방법이 있다. IRDS 명령어를 주 언어에 삽입하는 방법은 원천 프로그램을 주 언어 프로그램으로 선행처리(precompile)하는 선행처리기(precompiler)가 필요하다. 선행처리된 중간 프로그램(immediate program)은 삽입된 IRDS 명령어를 처리하는 라이브러리 함수를 호출하는 프로그램이기 때문에 이러한 인터페이스 방법을 사용하기 위해서는 먼저 CLI(Call Level Interface) 형태의 라이브러리 함수들이 구현되어야 한다. 본 연구에서는 정보저장소 인터페이스로 주 언어를 C로 하고 C 프로그램에서 라이브러리 함수로 호출할 수 있는 CLI를 구현하였다.

본 논문의 구성은 다음과 같다. 1.1절에서는 IRDS의 자료 구조에 대하여 살펴보고, 1.2절에서는 IRDS를 중심으로한 정보저장소에 대한 표준화 현황을 설명하며 1.3 절에서는 ANSI IRDS 표준안에 근거하여 구현된 IRDS 서비스 인터페이스 프로토타입에 대한 구조를 설명한다. 2장에서는 IRDS C CLI가 제공하여야 할 요구사항을 정의하고, 3장에서는 정의된 IRDS C CLI의 요구 사항을 지원하기 위해서 필요한 자료 구조와 구현 설계를 설명하였다. 4장에서는 3장에서 설계된 IRDS C CLI을 토대로 하여 구현된 내용에 대하여 설명하였다. 끝으로 5장에서는 본 연구의 결론을 맺고, 향후 연구할 내용을 기술하였다.

1.1 IRDS 자료 구조

IRDS의 자료 구조는 형(type)과 인스턴스(instance)라는 개념에 기초하고 있다. IRDS 구조의 핵심은 데이터 추상화 정도에 따른 4개의 계층에 있다. 4개의 계층은 형과 인스턴스의 관계에 있고 (그림 1)에서 보는 바와 같이 IRD 스키마 설명 계층은 IRD 스키마 계층에 저장될 정보의 형을 저장하고 IRD 스키마 계층은 IRD 스키마 설명 계층에 저장된 형의 인스턴스를 저장하면서 동시에 IRD 계층에 저장될 인스턴스의 형을 저장한다. 그리고 IRD 계층은 IRD 스키마



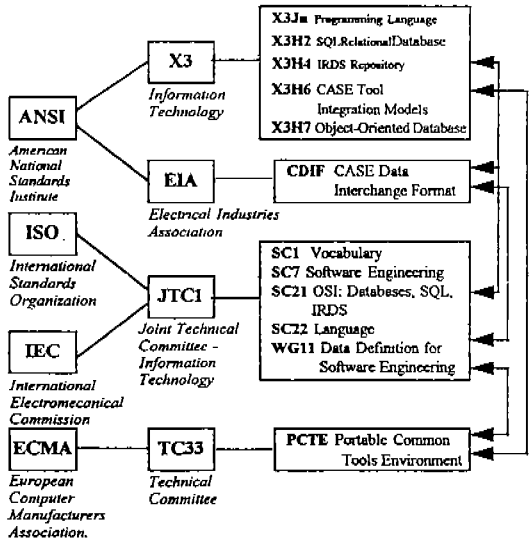
(그림 1) IRDS 자료 구조
(Fig. 1) Data Structure of IRDS

계층에 저장된 형에 대한 인스턴스를 저장하면서 동시에 응용 계층에 저장되는 정보의 형을 저장한다. 그러므로 이러한 계층들 간에는 내포(intension)와 외연(extension)의 관계가 성립한다[2, 17, 18].

이러한 형과 인스턴스 개념은 오늘날 응용 프로그램이나 DBMS에서 특정 자료에 접근하기 전에 접근하려는 자료의 형이 무엇인지를 결정하고 자료 처리를 하는 것과 동일한 개념으로 설계되었다. (그림 1)에 나타난 IRDS 자료 구조의 4개의 계층에 대한 역할은 다음과 같다.

- IRD 스키마 설명(IRD Schema Description) 계층
  이 계층은 IRD 스키마 계층에 저장될 객체에 대한 형을 정의하여 IRD 안에 저장되는 데이터에 대한 존재론적 구조를 설명한다. IRD 계층에 저장될 정보는 반드시 이 계층으로부터 기술해야 한다.
- IRD 스키마(IRD Schema) 계층
  이 계층은 IRD 스키마를 저장하고, 하나 이상의 IRD 스키마를 정의할 수 있으며 이들은 반드시 IRD 스키마 설명 계층에서 정의되어 있어야 한다. IRD 스키마는 IRD 계층에 정의된 데이터의 형과 IRD 계층 관리에 필요한 제어 정보를 포함한다.
- IRD 계층
  이 계층은 응용 계층에 저장되는 데이터에 대한 형을 저장한다. 또한, 응용 계층에 저장된 데이터를 관리하는 응용 프로그램에 관한 분석, 설계, 제어에 관련된 정보와 버전, 형상관리에 필요한 정보 등을 포함한다. 이 계층에는 많은 양의 IRD 들이 존재할 수 있으며 이들은 하나의 IRD 스키마에 의해 정의되어야만 한다.
- 실 계층
  이 계층은 IRD 계층에서 정의한 데이터의 인스턴스가 저장되고 데이터의 종류는 정보 시스템의 사용자가 다루는 일반 데이터이다. 따라서 응용 프로그램 개발자들에게 관심이 있는 부분이며 이 응용 계층에 있는 데이터를 접근하기 전에 우선 IRD에 저장되어 있는 메타데이터를 참조하게 된다.

IRDS는 4개의 IRDS 자료 계층 중에서 IRD 스키마 설명 계층, IRD 스키마 계층, 그리고 IRD 계층을 관리하고 응용 계층은 DBMS에 의해 관리된다. 따라서 IRDS는 관리하려 하는 데이터의 명세만을 관리하는 시스템이다.



(그림 2) CASE 표준화 기구 및 위원회
(Fig. 2) CASE Standardization Bodies and Committees

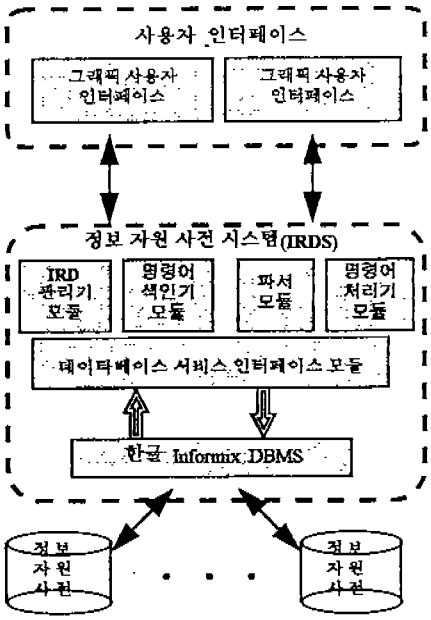
1.2 CASE 정보저장소의 표준화 현황

국제적인 표준화 단체들은 CASE 정보저장소의 표준화를 위해 많은 노력을 기울이고 있다. 이러한 표준화 노력은 (그림 2)에서 나타난 바와 같이 미국 국립 표준국(American National Standards Institute: ANSI), 국제 표준화 기구(International Organization for Standardization: ISO), 국제 전기 표준 회의(International Electrotechnical Commission: IEC), 그리고 유럽 전자계산기 공업회(European Computer Manufacturers Association: ECMA)를 중심으로 이루어지고 있다[11]. 이러한 표준화 노력으로 미국 국립 표준국의 정보처리 표준화 위원회 X3 산하 기술 위원회 H4와 국제 표준화 기구와 국제 전기 표준 회의의 정보 처리 연합 위원회 산하 SC21 기술 위원회에서 IRDS가 정보저장소의 표준으로 채택되었다. 국제 표준화 기구의 IRDS 표준은 SQL(Structured Query Language)에 바탕을 둔 정보 모델링을 택하고 있고

미국 국립 표준국의 IRDS 표준은 개체-관계(entity-relationship) 정보 모델링을 택하고 있다.

1.3 IRDS 서비스 인터페이스 프로토타입

ANSI IRDS 표준안에 근거하여 구현된 IRDS의 논리적 구조는 (그림 3)과 같이 하나의 자립 시스템(stand-alone)으로 구현되었다[15, 19]. (그림 3)에 나타난 IRDS는 한글 Informix Online 엔진을 하부 구조로 하여 IRD를 관리하고 사용자로부터 모든 IRDS 명령어를 입력받아 그 처리 결과를 화면으로 보여 준다.



(그림 3) IRDS 서비스 인터페이스 전체 구조
(Fig. 3) Overall structure of IRDS service interface

(그림 3)에 나타난 IRDS는 사용자 인터페이스로 대화식 사용자 인터페이스와 그래픽컬 사용자 인터페이스를 제공한다. 대화식 사용자 인터페이스는 모든 IRDS 명령어를 텍스트 입력 화면에서 입력하고 그 결과를 텍스트 출력 화면에서 확인할 수 있다. 그래픽컬 사용자 인터페이스는 X11R4 OSF/Motif 1.1을 바탕으로 하여 〈표 1〉과 같은 IRDS 명령어 중에서 예약어(reserved word)를 제외한 식별자(identifier) 만을 입력하고 그 처리 결과를 제공한다. (그림 3)의 IRDS는 어떠한 프로그래밍 언어와의 주 언어 인터페이스도 지원하지 않는다.

2. IRDS C CLI의 필요성 및 요구 사항

(그림 3)에 나타난 IRDS는 모든 IRDS 명령어를 사용자 인터페이스를 통하여 입력받고 입력된 명령어들의 처리 결과를 모두 사용자 인터페이스로 출력한다. 다시 말하여 모든 IRDS 명령어를 키보드로 입력하고 모든 명령어들의 처리 결과를 화면으로 출력한다. 이러한 명령어 입출력 방법은 CASE 도구의 정보저장소로서 IRDS를 사용하기에는 매우 부적합하다. 따라서 CASE 도구와 IRDS 간의 정보저장소 인터페이스 역할을 하는 IRDS C CLI는 새로운 명령어 입출력 방법이 필요하다.

IRDS C CLI는 C 프로그램에서 IRDS의 서비스가 필요할 때 IRDS에게 IRDS 명령어를 전달하고 그 결과를 C 프로그램으로 돌려줄 수 있도록 하기 위하여 다음과 같은 요구 사항이 필요하다.

- IRDS 명령어 전달 기능
- IRDS 명령어 실행 결과 관리 기능
- IRDS 명령어 실행 판단 기능

〈표 1〉 IRDS 명령어의 종류
〈Table 1〉 Kinds of IRDS Commands

IRD 명령어	IRD 스키마 명령어	유틸리티 명령어
ADD ENTITY	ADD META-ENTITY	CREATE IRD
MODIFY ENTITY	MODIFY META-ENTITY	REMOVE IRD
DELETE ENTITY	DELETE META-ENTITY	
ADD RELATIONSHIP	ADD META-RELATIONSHIP	
MODIFY RELATIONSHIP	MODIFY META-RELATIONSHIP	
DELETE RELATIONSHIP	DELETE META-RELATIONSHIP	
MODIFY ENTITY ACCESS-NAME	MODIFY META-ENTITY	
MODIFY ENTITY	ACCESS-NAME	
DESCRIPTIVE-NAME	OUTPUT IRD-SCHEMA	
COPY ENTITY		
OUTPUT IRD		

2.1 IRDS 명령어 전달 기능

이 기능은 C 프로그램에서 IRDS에게 원하는

서비스를 전달하기 위해서 필요하다. IRDS C CLI는 〈표 1〉과 같은 IRDS 명령어들을 전달할 수 있어야 한다. 〈표 1〉에 나타난 IRDS 명령어들은 [1]과 [10]에서 정의된 명령어들 중에서 IRD간의 Import/Export 명령을 제외한 것을 나타낸다. [19]에서 구현한 IRDS 서비스 인터페이스 프로토타입 시스템은 〈표 1〉에 나타난 명령어들을 실행할 수 있다.

2.2 IRDS 명령어 처리 결과 저장 기능

이 기능은 C 프로그램에서 IRDS에 의하여 수행된 IRDS 명령어의 처리 결과를 관리할 수 있도록 하기 위해서 필요하다. 〈표 1〉에 나타난 IRDS 명령어들 중에서 검색 명령어에 속하는 "output ird" 명령어와 "output ird-schema" 명령어에 의해 검색된 메타 정보나 메타 스키마 정보를 저장할 수 있어야 한다. "output"에 관련된 IRDS 명령어는 다양하게 검색 조건을 줄 수 있기 때문에 IRDS C CLI는 각각의 검색 조건에 맞는 처리 결과를 저장하고 관리하여야 한다.

2.3 IRDS 명령어 실행 판단 기능

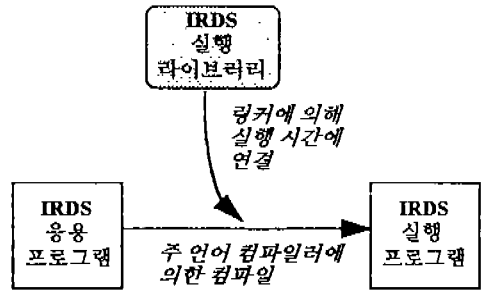
이 기능은 C 프로그램에서 IRDS 명령어가 성공적으로 완료되었는지, 혹은 실행 도중 오류가 발생했는지를 확인하기 위해서 필요하다. 만약 오류가 발생했다면 IRDS C CLI는 오류 원인을 사용자에게 알려줄 수 있어야 한다.

3. IRDS C CLI의 설계

본 장에서는 정보저장소 인터페이스 방식을 살펴보고, IRDS C CLI의 요구 사항을 충족시키기 위한 사항들을 설계한다.

3.1 IRDS의 정보저장소 인터페이스 방법

IRDS의 정보저장소 인터페이스 방식은 크게 두 가지로 분류할 수 있다[5]. 첫 번째 방법은 주 언어에서 호출하여 연결기(linker)에 의해 실행 시간에 연결(link)되는 라이브러리 함수들의 인수로 IRDS 명령어를 전달하는 방법이다. 이러한 방법을 사용했을 때 원시 화일에서 실행 화일을 생성시키는 과정이 (그림 4)에 나타나 있다.



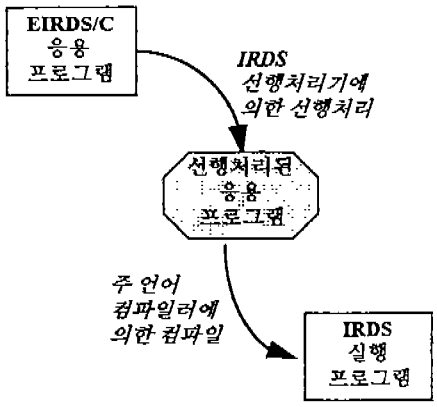
(그림 4) IRDS 응용 프로그램 컴파일 과정
(Fig. 4) Compiling Process of IRDS application programs

이 방법은 IRDS 명령어를 C 프로그램에 삽입하여 선행처리 과정을 거치도록 하기 위해서 반드시 필요하다. 이러한 인터페이스 방법의 대표적인 예는 Oracle 관계형 DBMS에서 제공하는 OCI(Oracle Call Interface)가 있다. OCI는 주 언어에서 라이브러리 함수로 호출하여 Oracle 데이터베이스를 처리할 수 있도록 해주는 라이브러리 함수들의 집합이다.

두 번째 방법은 주 언어에 IRDS 명령어를 삽입하여 선행처리기에 의해 주 언어에 맞도록 원천 프로그램을 선행처리한 후 이를 다시 주 언어 컴파일러를 이용해서 실행 코드를 생성시키는 방법이다. 이러한 방법을 사용했을 때 원시 프로그램에서 실행 프로그램을 생성시키는 과정이 (그림 5)에 나타나 있다.

이러한 방법의 예는 DBMS의 한 모듈로서 제공되는 ESQL/C(Embedded Structured Query Language with C) 선행처리기를 이용하여 주 언어인 C 프로그램에 SQL를 삽입하여 응용 프로그램을 작성하는 ESQL/C 프로그램이 있다. 이 방법의 장점은 SQL을 직접 C 프로그램에 내포시킬 수 있기 때문에 SQL을 아는 사람이라면 비교적 쉽게 응용 프로그램을 개발할 수 있다. 그러나 이를 위해서 선행처리기가 반드시 필요하다. 본 연구에서는 일차적으로 정보저장소 인터페이스로서 첫 번째 방법을 따르는 IRDS C CLI를 구현하였다. 본 연구에서 구현한 라이브러리

함수들은 향후 두 번째 방법과 같은 정보저장소 인터페이스 방식을 제공하려 할 때 선행처리기에 의해 호출될 수 있다.



(그림 5) EIRDS 응용 프로그램 컴파일 과정
(Fig. 5) Compiling Process of EIRDS application program

3.2 IRDS 명령어 전달 기능 설계

IRDS 명령어를 가리키는 문자열 포인터 변수를 IRDS C CLI 라이브러리 함수의 인수로 전달하여 IRDS 명령어를 전달한다. IRDS 명령어는 프로그램이 컴파일 되면서 정적으로 완전하게 바인딩될 수도 있고 프로그램이 실행되면서 개발자에 의해 동적으로 바인딩될 수도 있다.문자열을 가리키는 포인터 변수는 C 언어에서 제공하는 표준 라이브러리 함수들 중에서 문자열을 관리하는 라이브러리 함수의 인수로 전달할 수 있으므로 IRDS 명령어의 전달 기능을 효율적으로 처리해 줄 수 있다. 또한 CASE 도구와 결합하여 하나의 프로세스로 사용할 수 있도록 할 때 간단하게 사용할 수 있다.

3.3 IRDS 명령어 처리 결과 저장 기능 설계

IRDS C CLI를 통해 전달되는 IRDS 명령어들 중에서 IRD에 저장된 메타 정보와 메타 스키마 정보를 검색하는 명령어는 "output ird" 명령어와 "output ird-schema" 명령이 있다. 이외의 IRDS 명령어는 IRD에 메타 정보나 메타 스키마 정보를 삽입하거나 삭제하거나 혹은 수정하는 명령어들과 새로운 IRD를 생성하거나 삭제하는 명령어들이다. 따라서 "output ird" 명령어와 "output ird-schema" 명령을 제외한 명령어들은 검색 결과를 저장하기 위한 특별한 자료구조가 필요하지 않다.

〈표 2〉는 "output ird select entities access-name = emp-account-no show all;" 명령에 의해 처리된 결과를 나타낸다. 이러한 검색 결과를 주 언어에서 처리할 수 있도록 하기 위해서 (그림 6)과 같은 자료 구조를 설계하였다. 검색 결과는 일반적으로 하나 이상의 튜플이기 때문에 연결 리스트(linked list)로 저장하기 위하여 다음 구조체를 가리키는 포인터 변수를 구조체의 구성원(member)으로 선언한다.

〈표 2〉 "output ird" 명령에 의한 검색 결과의 예
〈Table 2〉 Example of the results of "output ird" command

```
                    Entity
Entity = EMP-ACCOUNT-NO
Entity Descriptive Name = EMPLOYEE-ACCOUNT-NUMBER
Entity-Type = ELEMENT
Description = Employee's account number with a financial
              institution, such as a number identifying a
              checking account at a bank
                   Attributes
Added-by = Joong-Ki Park
Last-Modified-By = Joong-Ki Park
Number-of-Times-Modified = 1
Source = PAYROLL-DEPT
                Attribute-Groups
data-time-added
   system-data = 19941120
   system-time = 092516
data-time-last-modified
   system-data = 19941220
   system-time = 141021
                  Relationships
PAYMENT-FINANCIAL-INST REPORT-CONTAINS-ELEMENT EMP-ACCOUNT-NO
```

(그림 6)과 (그림 7)의 자료 구조는 IRDS C CLI를 이용하는 응용 프로그램 개발자들에게 개방되는 자료구조이며 개발자는 이러한 자료구조를 가리키는 포인터 변수를 IRDS 명령어를 가리키는 문자열 변수와 함께 검색 기능을 갖는 IRDS C CLI 라이브러리 함수의 형식 매개 변수(formal parameter)로 전달하여 IRDS 엔진에 의해 처리된 검색 결과를 C 프로그램으로 받아온다.

```
struct outputda
{
    char *etype;
    char *ename;
    char *edname;
    char *rel[RELNUM];
    struct outird *next;
};
```

(그림 6) "output ird" 명령을 위한 자료 구조
(Fig. 6) Data structure for "output ird" command

〈표 3〉은 "output ird-schema select document show all;" 명령에 의해 처리된 검색 결과를 나타내고 (그림 7)은 이러한 명령에 의해 검색되는 메타 스키마 정보를 저장하기 위한 자료 구조를 나타낸다. 〈표 2〉는 개체(entity) "EMP-ACCOUNT-NO"에 관련된 모든 정보를 나타내주고 있다. (그림 3)에 나타난 바와 같이 자립 시스템으로 구현된 IRDS는 〈표 2〉과 같은 명령어의 처리 결과가 화면으로 출력이 된다.

〈표 3〉 "output ird-schema" 명령에 의한 검색 결과의 예
〈Table 3〉 Example of results of "output ird-schema" command

```
                    Meta-Entity
Meta-Entity = DOCUMENT
Meta-Entity-Type = Entity-Type
                  Meta-Attributes
Added-By = BASIC-FUNCTIOAL-SCHEMA
Meta-Entity-Substitute-Name = DOC
Connectable = YES
              Meta-Attribute-Groups
date-time-added
   system-date = 19941220
   system-time = 152654
date-time-last-modified
   system-date = 19941210
   system-time = 093150
                  Meta-Relationships
DOCUMENT ENTITY-TYPE-CONTAINS-ATTRIBUTE-TYPE ADDED-BY
```

3.4 IRDS 명령어 실행 판단 기능 설계

명령어 전달 변수를 통해 전달된 IRDS 명령어가 성공적으로 실행이 완료됐는지 혹은 오류가 발생했는지를 파악하기 위하여 명령어 실행 판단 변수를 둔다. 이러한 명령어 판단 변수를 사용하여 하나의 IRDS 명령어를 실행할 때마다 명령어의 실행 판단을 할 수 있도록 한다. IRDS C CLI는 하나의 IRDS 명령어를 실행할 때마다 실행 코드를 명령어 실행 판단 변수에 저장한다.

```
struct outshada
{
    char *metype;
    char *mename;
    char *mrel[RELNUM];
    struct outshada *next;
};
```

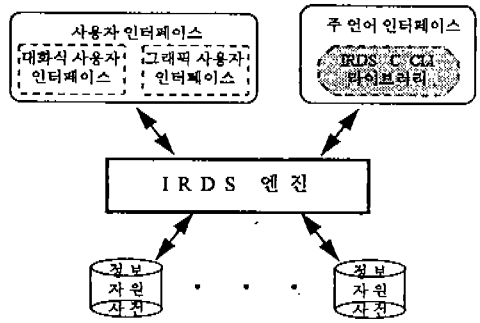
(그림 7) "output ird-schema" 명령에 의한 검색 결과 저장 구조
(Fig. 7) Data structure to store the results of "output ird-schema" command

4. IRDS C CLI의 구현

본 장에서는 IRDS C CLI을 구현한 시스템 환경과 IRDS C CLI 요구사항을 지원하기 위해서 설계한 것을 토대로 구현한 IRDS C CLI을 구성하는 라이브러리 함수들에 대한 설명을 한다 [16]. (그림 8)은 주 언어 인터페이스로 본 논문에서 구현한 IRDS C CLI와 기존에 구현된 IRDS 프로토타입과의 관계를 나타낸다.

4.1 IRDS C CLI의 개발 환경

IRDS C CLI는 Sun SPARC 워크스테이션의 SunOS 4.1.3 상에서 한글 Informix OnLine ESQL/C 프로그램으로 개발되었다. 사용된 선행



(그림 8) 주 언어 인터페이스를 지원하는 IRDS 전체 구조
(Fig. 8) Overall structure of IRDS supporting host language interface

처리기는 한글 Informix OnLine의 선행처리기를 사용하였고 C언어로 선행처리된 원시 프로그램의 컴파일러로서 SunOS에서 제공하는 ANSI C 컴파일러와 호환성을 갖춘 C 컴파일러를 사용하였다.

4.2 IRDS 명령어 전달 기능 구현

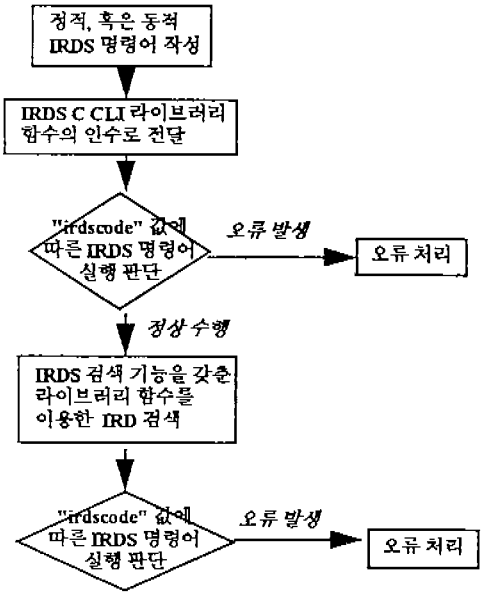
IRDS 명령어를 가리키는 문자열 변수를 IRDS C CLI를 구성하는 라이브러리 함수의 인수로 전달하도록 하였다. 이러한 문자열 변수는 주 언어인 C로 작성되는 프로그램에서 개발자에 의해 선언이 되고 선언된 문자열 포인터 변수에 IRDS 명령어가 정적으로 혹은 동적으로 바인딩되어 IRDS C CLI를 구성하는 라이브러리 함수를 호출할 때 인수로 전달된다. 이러한 인수는 IRDS C CLI를 구성하는 라이브러리 함수에서 "irdscmd"라는 형식 매개변수(formal parameter)에 저장되며, 매개 변수 전달은 참조 호출(call by reference)로 이루어진다.

4.3 IRDS 명령어 처리 결과 저장 기능 구현

IRDS 명령어들 중에서 검색 명령어인 "output ird" 명령과 "output ird-schema" 명령의 검색 결과를 저장하기 위하여 (그림 6)과 (그림 7)에 나타난 두개의 자료 구조를 사용한다. 개발자는 C 프로그램에서 필요에 따라 (그림 6)이나 (그림 7)의 구조체 형을 갖는 변수를 선언한 후 IRDS 검색 명령어와 함께 IRDS C CLI에게 전달한다. 검색 결과를 저장하기 위한 변수는 참조 호출 방식을 따르는 "result"와 "schres"라는 형식 매개변수이며, IRDS C CLI는 처리된 검색 결과를 이 형식 매개변수에 저장한다.

4.4 IRDS 명령어 실행 판단 기능 구현

IRDS C CLI에 전달된 IRDS 명령어의 실행 판단을 위하여 오류 원인(error source)에 따른 오류 코드를 구성하여 전역 변수인 "irdscode"에 저장한다. IRDS C CLI는 IRDS에 의해 보고되는 모든 IRDS 명령어의 실행 코드를 전역 변수 "irdscode"에 저장한다. 전역 변수 "irdscode" 값에 따라 IRDS 명령어 실행 판단을 할 수 있다. 다음은 "irdscode"에 저장되는 값의 의미를 나타낸다.



(그림 9) IRDS C CLI를 이용한 응용 프로그램 흐름
(Fig. 9) Control flow of application program using IRDS C CLI

- IRDS 오류
  - 문법 오류 : −5000~−6000
  - 명령어 의미 오류 : −7000~−8000
  - 실행 시간 오류 : −8000~−9000
- 한글Informix ESQL/C 오류 : −100~−1300
- 성공적인 명령어 실행 : 0보다 큰 값

4.5 IRDS C CLI 라이브러리 함수들의 기능

IRDS C CLI를 구성하는 라이브러리 함수는 공통적으로 "Irds" 시작하며 IRD 명령어와 IRD 스키마 명령어, 그리고 유틸리티 명령을 합쳐 모두 20개로 구성된다. (그림 9)는 C 프로그램에서 IRDS C CLI를 호출하여 IRD를 접근할 때 프로그램의 흐름을 나타낸다. (표 4)는 IRDS C CLI를 구성하는 라이브러리 함수들을 기능별로 분류한 것을 나타낸다.

(1) IrdsAddEnt(char *irdscmd)

IRD 명령어 처리 함수로서 IRDS 명령어 중에 "add entity"로 시작하는 IRD 명령어를 수행하는 함수이다. 이 함수가 수행하는 IRDS 명령어는 질의 결과가 있는 검색 명령어가 아니기 때문에 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료 구조는 필요하지 않으며 명령어의 실행 판단을 위하여 전역 변수인 "irdscode"에 실행 결과를 저장한다. "irdscode"에 저장된 정수 값이 양수이면 전달된 "add entity" 명령어가 성공적으로 수행됐음을 알려 주고 음수이면 전달된 명령어에 오류가 있었음을 알려 준다. 오류 코드는 오류 원인에 따라 IRDS 오류, 한글 Informix 오류로 분류된다.

〈표 4〉 IRDS C CLI를 구성하는 라이브러리 함수들의 종류
〈Table 4〉 Sorts of library functions composed of IRDS C CLI

IRD 처리 CLI	IRD 검색 CLI	유틸리티 CLI
IrdsAddEnt(), IrdsModEnt(), IrdsDelEnt(), IrdsAddRel(), IrdsModRel(), IrdsModName(), IrdsModDname(),IrdsCopyEnt(), IrdsDelRel(), IrdsAddMenty(), IrdsModMenty(), IrdsModMrel(), IrdsDelMenty(), IrdsAddMrel(), IrdsDelMrel(), IrdsModMname(),	IrdsOutDict(), IrdsOutSchema()	IrdsCreDict(), IrdsRemDict()

(2)IrdsModEnt(char *irdscmd)

IRD 명령어 처리 함수로서 IRDS 명령어 중에 "modify entity"로 시작하는 IRD 명령어를 수행하는 함수이다. 이 함수가 수행하는 IRDS 명령어는 질의 결과가 있는 검색 명령어가 아니므로 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료 구조는 필요하지 않으며, 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용한다.

(3) IrdsDelEnt(char *irdscmd)

IRD 명령어 처리 함수로서 IRDS 명령어 중에 "delete entity"로 시작하는 IRD 명령어를 수행하는 함수이다. 이 함수가 수행하는 IRDS 명령어는 질의 결과가 있는 검색 명령어가 아니므로 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료 구조는 필요하지 않으며, 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용한다.

(4) IrdsAddRel(char *irdscmd)

IRD 명령어 처리 함수로서 IRDS 명령어 중에 "add relationship"으로 시작하는 IRD 명령어를 수행하는 함수이다. 이 함수가 수행하는 IRDS 명령어는 질의 결과가 있는 검색 명령어가 아니므로 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료 구조는 필요하지 않고 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용한다.

(5) IrdsModRel(char *irdscmd)

IRD 명령어 처리 함수로서 IRDS 명령어 중에 "modify relationship"으로 시작하는 IRD 명령어를 수행하는 함수이다. 이 함수가 수행하는 IRDS 명령어는 질의 결과가 있는 검색 명령어가 아니므로 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료 구조는 필요하지 않고 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용한다.

(6) IrdsDelRel(chr *irdscmd)

IRD 명령어 처리 함수로서 IRDS 명령어 중에 "delete relationship"으로 시작하는 IRD 명령어를 수행하는 함수이다. 이 함수가 수행하는 IRDS 명령어는 질의 결과가 있는 검색 명령어가 아니므로 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료 구조는 필요하지 않고 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용한다.

(7) IrdsModName(char *irdscmd)

IRD 명령어 처리 함수로서 IRDS 명령어 중에 "modify entity access-name"으로 시작하는 IRD 명령어를 수행하는 함수이다. 이 함수가 수행하는 IRDS 명령어는 질의 결과가 있는 검색 명령어가 아니므로 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료구조는 필요하지 않지만 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용한다.

(8) IrdsModDname(char *irdscmd)

IRD 명령어 처리 함수로서 IRDS 명령어 중에 "modify entity descrip tive-name"으로 시작하는 IRD 명령어를 수행하는 함수이다. 이 함수가 수행하는 IRDS 명령어는 질의 결과가 있는 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료 구조는 필요하지 않지만 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용한다.

(9) IrdsCopyEnt(char *irdscmd)

IRD 명령어 처리 함수로서 IRDS 명령어 중에 "copy entity"로 시작하는 IRD 명령어를 수행하는 함수이다. 이 함수가 수행하는 IRDS 명령어는 질의 결과가 있는 검색 명령어가 아니므로 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료 구조는 필요하지 않지만 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용한다.

(10) IrdsOutDict(char *irdscmd, struct outputda *result)

IRD 명령어 처리 함수로서 IRDS 명령어 중에 "output ird"로 시작하는 IRD 명령어를 수행하는 함수이다. "output ird" 명령은 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용하며 IRD에 저장되어 있는 메타 정보 중에서 원하는 정보를 추출하는 검색 명령이기 때문에 명령어 처리 결과를 C 프로그램으로 넘겨줄 수 있어야 한다. 이를 위해서 4장에서 검색 결과를 저장하기 위한 자료 구조를 설계하였고 이러한 자료 구조를 가리키는 "result"형식 매개변수를 이용하여 "output ird" 명령의 검색 결과를 저장한다.

(11) IrdsAddMenty(char *irdscmd)

IRD 스키마 명령어 처리 함수로서 IRDS 명령어 중에 "add meta-entity"로 시작하는 IRD 스키마 명령어를 수행하는 함수이다. 이 함수가 수행하는 명령어는 질의 결과가 있는 검색 명령어가 아니므로 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료구조는 필요하지 않지만 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용한다.

(12) IrdsModMenty(char *irdscmd)

IRD 스키마 명령어 처리 함수로서 IRDS 명령어 중에 "modify meta-entity"로 시작하는 IRD 스키마 명령어를 수행하는 함수이다. 이 함수가 수행하는 명령어는 질의 결과가 있는 검색 명령어가 아니므로 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료구조는 필요하지 않지만 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용한다.

(13) IrdsDelMenty(char *irdscmd)

IRD 스키마 명령어 처리 함수로서 IRDS 명령어 중에 "delete meta-entity"로 시작하는 IRD 스키마 명령어를 수행하는 함수이다. 이 함수가 수행하는 명령어는 질의 결과가 있는 검색 명령어가 아니므로 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료구조는 필요하지 않지만 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용한다.

(14) IrdsAddMrel(char *irdscmd)

IRD 스키마 명령어 처리 함수로서 IRDS 명령어 중에 "add meta-relationship"으로 시작하는 IRD 스키마 명령어를 수행하는 함수이다. 이 함수가 수행하는 명령어는 질의 결과가 있는 검색 명령어가 아니므로 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료 구조는 필요하지 않지만 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용한다.

(15) IrdsModMrel(char *irdscmd)

IRD 스키마 명령어 처리 함수로서 IRDS 명령어 중에 "modify meta-relationship"으로 시작하는 IRD 스키마 명령어를 수행하는 함수이다. 이 함수가 수행하는 명령어는 질의 결과가 있는 검색 명령어가 아니므로 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료 구조는 필요하지 않지만 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용한다.

(16) IrdsDelMrel(char *irdscmd)

IRD 스키마 명령어 처리 함수로서 IRDS 명령어 중에 "delete meta-relationship"으로 시작하는 IRD 스키마 명령어를 수행하는 함수이다. 이

함수가 수행하는 명령어는 질의 결과가 있는 검색 명령어가 아니므로 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료 구조는 필요하지 않지만 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용한다.

(17) IrdsModMname(char *irdscmd)

IRD 스키마 명령어 처리 함수로서 IRDS 명령어 중에 "modify meta-entity access-name"으로 시작하는 IRD 스키마 명령어를 수행하는 함수이다. 이 함수가 수행하는 명령어는 질의 결과가 있는 검색 명령어가 아니므로 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료 구조는 필요하지 않지만 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용한다.

(18) IrdsOutSchema(char *irdscmd, struct outschda *schres)

IRD 스키마 명령어 처리 함수로서 IRDS 명령어 중에 "output ird-schema"로 시작하는 IRD 명령어를 수행하는 함수이다. "output ird-schema" 명령은 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용하며 IRD에 저장되어 있는 메타 스키마 정보 중에서 원하는 정보를 추출하는 검색 명령이기 때문에 명령어 처리 결과를 C 프로그램으로 넘겨줄 수 있어야 한다. 이를 위해서 4장에서 검색 결과를 저장하기 위한 자료 구조를 설계하였고 이를 가리키는 "schres"라는 형식 매개변수에 "output ird-schema" 명령의 검색 결과를 저장한다.

(19) IrdsCreDict(char *irdscmd)

IRD 유틸리티 명령어로서 IRDS 명령어 중에 "create ird"로 시작하는 IRDS 명령어를 수행하는 함수이다. 이 함수가 수행하는 명령어는 질의 결과가 있는 검색 명령어가 아니므로 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료 구조는 필요하지 않지만 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용한다.

(20) IrdsRemDict(char *irdscmd)

IRD 유틸리티 명령어로서 IRDS 명령어 중에 "remove ird"로 시작하는 IRDS 명령어를 수행하는 함수이다. 이 함수가 수행하는 명령어는 질의 결과가 있는 검색 명령어가 아니므로 명령어 처리 결과를 C 프로그램으로 넘겨주기 위한 자료 구조는 필요하지 않지만 명령어 실행 판단을 위하여 전역 변수 "irdscode" 값을 이용한다.

5. 결론 및 향후 연구 과제

본 논문에서는 구조적 분석 및 설계 방법론을 지원하는 CASE 도구의 표준 정보저장소로 사용하기 위해서 ANSI IRDS 표준안에 따라 이미 자립 시스템으로 구현된 IRDS를 위한 정보저장소 인터페이스로서 C 언어와의 주 언어 인터페이스를 구현하였다. C 프로그램에서 IRDS 엔진에 명령어를 전달하기 위해서 문자열 포인터 변수를 사용하였고 IRDS 검색 명령어의 처리 결과를 관리하기 위하여 명령어 처리 결과를 저장하는 구조체를 사용하였다. IRDS 명령어의 실행 판단을 위해서 각각의 명령어가 실행 될 때 마다 실행 결과 코드를 저장하는 명령어 실행 판단 변수를 사용하였다.

향후 연구 과제로는 IRDS C CLI를 통해 여러 사용자가 IRDS 엔진과 연결되어 실행될 때 발생할 수 있는 IRD 불일치성을 해결하기 위해 IRDS 엔진에 동시성 제어 기능을 보강하는 것이 요구된다. 더불어 IRDS의 가용도를 높이고 고성능의 IRDS 명령어 처리를 필요로 하는 IRDS 응용 프로그램 개발자를 위하여 API(Application Programming Interface)를 제공하는 것이 필요하다.

참 고 문 헌

[1] ANSI X3H4.1, The Information Resource Dictionary System(IRDS) Reference Model, X3H4/90-195R, May 1991.

[2] D. Altomare, et al, "A Prototype for the Integration of information Resource Dictionary System and PCTE," Computer Standards and Interface, 1989.

[3] Minder Chen, "A Framework for Integrated CASE," IEEE Software, March 1992.

[4] Daniel R. Dolk, et al "Relational Informa-

tion Resource Dictionary System," CACM, Vol. 30, No. 1, January 1987.
[5] James R. Groff, et al, "The SQL API Alternative," Database Programming and Design, November, 1990.
[6] Robert E. Hodges, Information Resource Dictionary Services Architecture Technical Report, X3H4/93-048R1, June 1993.
[7] ISO IS10027, Information Resource Dictionary System(IRDS) Framework, Rev. 9, 19th, January 1989.
[8] Mark R. Jones, "Brave New World: A Vision of IRDS," Database Programming and Design, November 1991.
[9] Mark R. Jones, "Unveiling Repository Technology," Database Programming and Design, April 1992.
[10] Magaret H. Law, Guide to Information Resource Dictionary System Applications: General Concepts and Strategic Systems Planning,NBS Special Publication 500-152.
[11] Dana M. Marks, "Unraveling the Standards," Database Programming and Design, December, 1993.
[12] Terry Moriatry, "Are You Ready for a Repository," Database Programming and Design, March, 1990.
[13] Mohan Prabandham, et al, "A View of the IRDS Reference Model," Database Programming and Design, March, 1990.
[14] Roger S. Pressman, Software Engineering: A Practitioner's Approach 3rd edition,McGraw-Hill, Inc., 1992.
[15] 김기봉, 박중기, 조유희, 진성일, "정보 자원 관리를 위한 IRDS 서비스 사용자 인터페이스의 설계 및 구현", 한국 정보 과학회 논문지 게재 심사중(논문 접수 번호: 94-12-10-280).
[16] 박중기, "정보 자원 사전 시스템을 위한 CCLI의 설계 및 구현", 석사 학위 논문, 충남대학교 자연과학 대학 컴퓨터 과학과, 1995.
[17] 정경업, "정보 자원 사전 시스템의 설계 및 구현," 석사학위 논문, 충남대학교 자연과학 대학 전산학과, 1993.
[18] 조완섭, 김명준, "IRDS," 정보통신 기술, Vol. 2, No. 3, Nov., 1988.
[19] 진성일, et al, 정보 자원 사전 시스템의 설계 및 구현, 시스템 공학 연구소, 최종 연구 보고서, 1994.

박 중 기

1993년 한남대학교 전자계산공학과 졸업(학사)
1995년 충남대학교 전산학과 대학원 졸업(이학석사)
1995~현재 시스템공학 연구소 소프트웨어 공학 연구부 연구원
관심분야 : 메타데이타베이스 시스템, 정보 자원 사전 시스템, 정보저장소 시스템

진 성 일

1978년 서울대학교 자연과대학 계산통계학과 졸업(학사)
1980년 한국과학기술원 전산학과 졸업(석사)
1994년 한국과학기술원 전산학과 졸업(박사)
1987년~89년 미국 Northwestern대학 전산학과 객원 교수
1990년~92년 충남대학교 전자계산소 소장
1980년~현재 충남대학교 자연과학대학 컴퓨터과학과 교수
1995년~현재 충남대학교 자연과학대학 컴퓨터과학과 학과장 .
관심분야 : 데이타베이스 시스템, 성능 분석, 정보 자원 사전 시스템, 객체 지향 모델링

이 헌 기

1987년 전북대학교 중어중문학과 졸업(학사)
1987년~현재 시스템공학 연구소 연구원
1991년 미국 IBM Sterling Forest연구소 객원연구원
관심분야 : 데이타모델링, 데이타베이스 시스템, 소프트웨어 공학, CASE