# 관계 DBMS를 이용한 정보 자원 사전 시스템(IRDS) 서비스 인터페이스 프로세서의 개발

조용호* · 이규철**

# Development of Services Interface Processor of information Resource Dictionary System based on the Relational DBMS

Jo Yong Ho, Lee Kyu Chul


## 요 약

컴퓨터의 각종 응용 시스템들이 다루는 데이타의 양과 복잡도가 높아짐에 따라 이를 효율적으로 다루기 위해 메타-데이타(meta-data)의 중요성이 날로 부각되고 있다. 하지만, 현재 DBMS의 기능은 조직체내의 데이타의 사용을 기술하고 제어하는 메타 데이타인 정보 자원(information resource)의 관리에는 그 기능이 미치지 못하고 있다.

최근들어, 이런 정보 자원들을 효율적으로 다루기 위해 차세대 정보 관리 시스템으로서 정보 자원 사전 시스템(Information Resource Dictionary System)이 제안되었다. 본 논문에서는 현재 표준화중인 정보 자원 사전 시스템을 분석하고, IRDS 서비스 인터페이스를 관계 DBMS를 기반으로 설계하고 구현하였으며, 이를 응용하여 그 효용성을 실험하였다.


## 1. 서 론

컴퓨터의 각종 응용 시스템들이 다루는 데이타의 양과 복잡도가 높아짐에 따라 이를 효율적으로 다루기 위해 데이타에 대한 데이타, 즉 메타 데이타(meta data)의 중요성이 날로 부각되고 있다. 예를 들어 화일 시스템의 디렉토리(directory)는 데이타를 담고 있는 각 화일의 접근 모드, 소유자, 크기, 형태등의 메타 데이타를 유지함으로써 화일을 효율적으로 다룰 수 있도록 도와준다.

근래에 들어 방대한 양의 데이타를 효율적으로 저장하고 관리하기 위해 많이 사용하는 데이타베이스 관리 시스템(Database Management System; DBMS)의 기능은 조직체내의

*다우기술 OSD 사업부
**충남대학교 공과대학 컴퓨터공학과
이 연구는 91년도 한국과학재단 연구비 지원에 의한 결과임(과제번호 913-1102-004-2)

인사 정보, 재고 정보, 회계 정보등의 단순 데이타 관리의 수준에 그치고 있으며, 이들 데이타의 사용이나 유지를 기술하고 제어하는 메타 데이타인 정보 자원(information resource)의 관리에는 그 기능이 미치지 못하고 있다[1][9][10].

이전의 데이타 사전(data dictionary)은 통상적으로 "데이타에 대한 데이타(meta-data)"란 개념에서 DBMS 패키지를 지원하기 위해 존재하였다. 그러나, 근래에 와서 데이타 사전들의 필요성이 소프트웨어의 요구 분석등 CASE(Computer-Aided Software Engineering) 기술에 크게 부각되고 있으며, 또한 분산 소프트웨어의 중요한 요소로서 이 환경에 필요한 모든 제어 정보에 대한 저장소로 사용되기 시작하였다. 데이타 사전 시스템은 전체 정보 조작을 제공할 수 있는 단계까지 도달했으며, 각 객체들은 화일, 데이타베이스, 컴퓨터, 네트워크, 사용자 추상화, 회사내 부서 및 재무 조직들과 같은 정보 자원들까지도 포함할 수 있게 되었다.

더우기 근래에는 의사 결정을 도와주는 전문가 시스템(expert system), 소프트웨어 생산성을 높이는 CASE 도구, 고 품질인 문서를 작성하는 탁상 출판 시스템(Desk-Top Publishing System)등의 각 응용 시스템들이 연계되면서 서로의 정보 자원을 공유하며 통합된 정보 시스템을 구축하는 경향이 있다. 따라서 조직체내의 각 응용 시스템들이 요구하고 사용하는 모든 정보 자원을 시스템간에 공유하고 통합 정의 및 관리하여 정보의 중복성을 제거하고 불일치성을 지양하는 시스템이 필요하게 되었는데(그림 1) 이것이 바로 정보 자원 사전 시스템(Information Resource Dictionary System; IRDS)이다[1].

IRDS는 정보 조작의 전체 과정을 표준화하는 것을 기본 목적으로 한다. 또한, IRDS의 기본 구조는 사용자에 의해 확장될 수 있도록 구성되어 있어서 독특한 환경 요구들을 받아들일 수 있도록 설계되어 있다.

본 논문은 위와 같은 필요성과 중요성을 지닌 IRDS의 기능을 응용 시스템들이 이용할 수 있도록 제공해주는 중요 요소인 IRDS 서비스 인터페이스 프로세서(Services Interface Processor)[5][6]를 세계 표준화 결과에 맞게 설계하고 구현하는데 그 목적을 두고 있다.

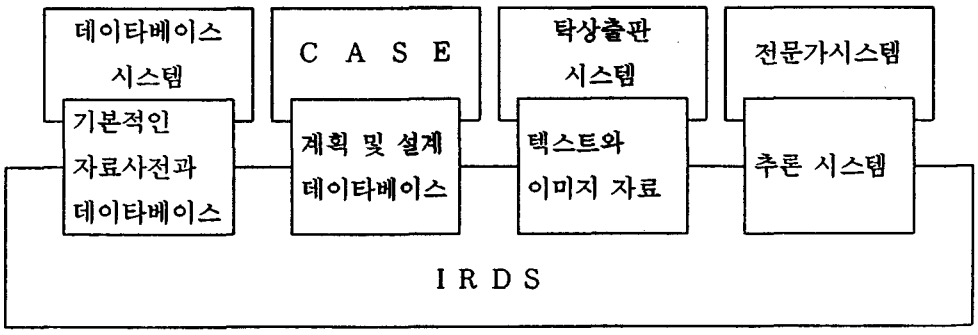



그림 1. 응용 시스템 통합을 위한 IRDS[1]

Fig. 1 IRDS for integration of application systems

본 논문의 구성은 다음과 같다. 2장에서 IRDS와 관련되어 진행되어 왔던 연구에 대해 살펴보고 IRDS의 차이점 및 장점을 기술하고, 3장에서 IRDS의 기본 구조 및 서비스 인터페이스에 대해 설명하였다. 4장에서는 IRDS 서비스 인터페이스 프로세서를 설계하여 구현한 사항에 대해 기술하고 5장에서는 구현된 결과를 응용하여 IRDS의 효용성을 알아 본다. 마지막으로 6장에서 결론과 추후 연구과제를 제시한다.

## Ⅱ. 관련 연구

IRDS 이전에 각 응용 시스템에서 독자적으로 사용되었던 유사한 시스템의 형태를 보면 데이타-요소 사전(data-element dictionary), DBMS 데이타 사전(DBMS data dictionary) 또는 카탈로그(catalog), 독립된 사전 시스템(stand-alone dictionary system), CASE 저장소(CASE repository)등을 들 수 있다[9].

1960년대 초, 많은 소프트웨어 응용에 관련된 데이타 요소들을 공통적으로 기술할 수 있는 사전의 생성 이점을 알기 시작했다. 개발자들은 종종 이들 데이타 요소와 데이타 요소에 대한 정의에 대해 타자기로 찍은 사전이나 천공 카드 사전을 만들었는데, 이것이 바로 초기 데이타-요소 사전이며, 오늘날에는 비록 형태는 다를지라도 같은 목적을 갖는 많은 사전 시스템들이 존재하고 있다[7].

1970년대 DBMS가 보급되었을 때, 데이타 정의의 제어 문제가 다시 발생하였으며, 이로 인해 생성된 사전 시스템이 DBMS 데이타 사전이다. 그러나, DBMS 데이타 사전은 외부 환경보다는 연산 및 사용을 지원하기 위한 것들을 더 중요하게 생각하였다.

또 다른 집합의 사전 기술 구현은 기본적으로 시스템, 소프트웨어 개발 그리고 문서를 지원하기 위한 독립적인 도구로서 발전하였다. 이런 도구들은 소프트웨어, 시스템, 데이타 그리고 조직체에 대한 정보의 저장소로 사용되었다. 이들 공급자에 의해 기술된 정의들은 구현된 소프트웨어가 사용하는 공통된 데이타 정의들로 부터 사전 시스템으로 만들어 졌다. 이들 시스템은 일반적으로 DBMS 공급자가 아닌 다른 공급자들에 의해 발전하였으며, 다양한 소프트웨어 지원 환경을 위한 교두보 역할을 했고, 또한 DBMS 데이타 사전보다는 폭 넓은 관점을 갖게 되었으며, 결국 이들 시스템은 IRDS의 원형이 되었다. 과거 몇년동안, 통상적인 CASE 도구들은 표준 발전에 커다란 영향을 끼쳤다. 이들은 특정한 소프트웨어와 시스템 공학 방법론에 기반하며, 이런 방법론에 필요한 정보들을 지원하기 위해 다양한 사전, 저장소 및 전문사전을 생성했다.

이러한 시스템들을 바탕으로, IRDS는 1980년대에 ANSI와 NBS(National Bureau of Standards)에서 사전 소프트웨어를 위한 표준을 발전시키기 시작했으며, 이들 노력에 의해 1983년에 ANSI X3H4의 IRDS 표준안이 ISO(International Organization for Standadization)에 제시되어, 1984년 1월에 ISO/IEC는 Subcommittee 21, Working Group 3(ISO TC97/SC21/WG3)에 국제 표준항목으로써 IRDS를 할당하여, 현재까지도 표준화에 대한 노력이 이루어지고 있다[7].

IRDS가 현재까지 사용되던 유사한 사전 시스템들에 비해 지니고 있는 장점으로는,

첫째, 어떤 특정 시스템에 종속적인 형태보다는 일반적으로 적용할 수 있는 구조 및 서비스를 제공하며,

둘째, 특정 도구 및 시스템에 국한하여 사용되는 정보보다는 한 기관의 응용 시스템들이 공통으로 사용하는 정보 자원을 관리하며,

셋째, 다단계(multilevel)의 메타 모델(meta model)을 지원하여 형태가 다른 모델링 을 따르는 여러 시스템을 통합하기에 적합하며,

넷째, 기존에 자신의 데이타 사전을 가지고 있는 시스템이라도 IRDS를 이용하면 추 가의 기능을 제공받을 수 있다. 예를 들어, DBMS 같은 경우에는 자신의 카탈 로그가 있으나 이는 데이타베이스 관리에만 사용되며, 응용을 개발할 때 버전 제어의 기능이 필요하다면 이는 IRDS를 사용하여 지원받을 수 있다.

따라서, IRDS는 한 기관에 존재하는 각 응용 시스템의 각종 정보 자원들을 효율적으로 관리하며, 이들 시스템을 통합하여 주는 기반 시스템으로 사용될 수 있다.

# Ⅲ. IRDS의 기본 구조 및 서비스 인터페이스

## 3.1 IRDS의 기본 구조

IRDS Framework표준[4]에 따르면, IRDS는 IRD(Information Resource Dictionary)와 IRDD (Information Resource Dictionary Definition)의 생성, 유지 및 접근을 위한 기능들을 제공하는 시스템이라고 정의하고 있다.

ISO IRDS 기본 구조는 4개의 단계(level)와 각 단계 사이의 단계쌍(level pair)으로 구성된다(그림 2)[4][7].

(1) IRDD 스키마 단계(IRD Definition Schema Level)
이 단계는 IRDD 단계에 저장될 데이타에 관련된 객체의 형을 기술한다.

(2) IRDD 단계(IRD Definition Level)
이 단계는 IRDD를 저장하기 위한 것이며, 하나 이상의 IRD에 저장될 수 있는 데이타에 관련된 객체의 형들을 기술하는 하나 이상의 IRD 스키마를 포함한다.

(3) IRD 단계(IRD Level)
이 단계는 IRD를 저장하기 위한 것이며, IRD 내용의 일부는 응용 단계에저장되는 데이타의 형을 정의할 수 있다.

(4) 응용 단계(Application Level)
이 단계는 일반 응용에 나타나는 데이타들을 저장한다.

IRD란 한 조직체의 전체 또는 일부에 관련된 정보 자원들의 정의를 위한 공유 가능한 저장 장소를 나타내며, 조직체에서 필요한 데이타와 이들 데이타를 유지하기 위한 과정들, 그와 같은 데이타들이 표현될 물리적인 하드웨어 환경, 정보를 사용할 수 있는 인적 물적 자원들의 조직 그리고 그 정보를 산출해 내는 인적 자원들의 전체 또는 일부에 관련된 정보를 포함한다. IRDD는 한 IRD내에 유지될 수 있는 데이타를 정의하는 객체들의 집합으로, 스키마의 부분이 아닌 데이타 또는 과거나 미래의

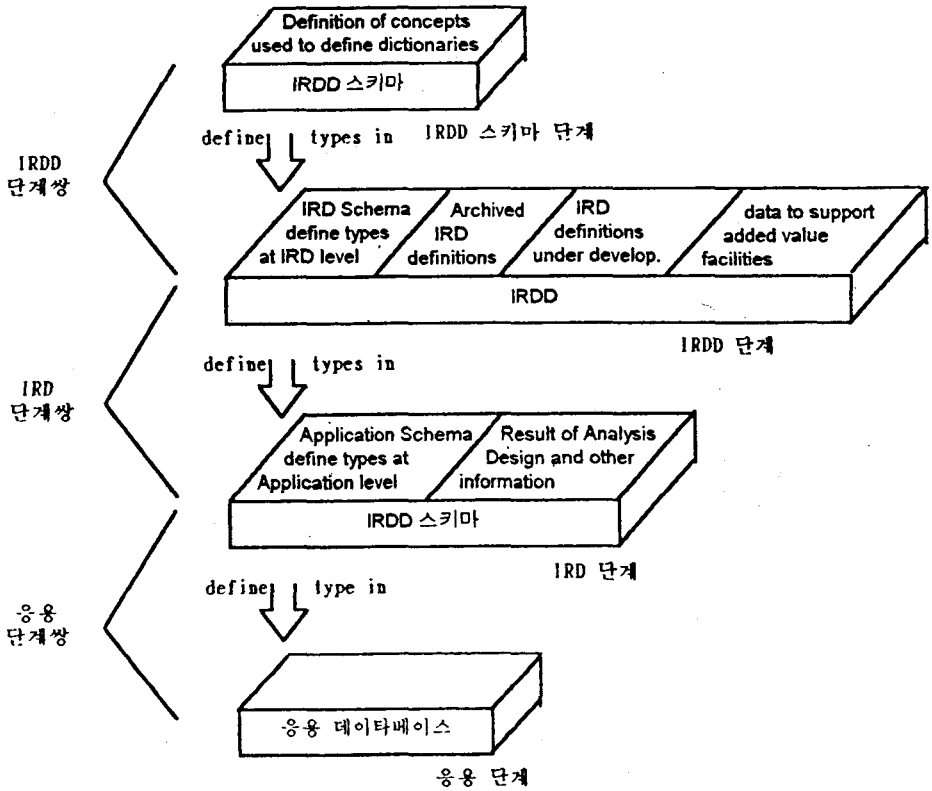



그림 2. IRDS의 단계들

Fig. 2 Levels of IRDS

IRD 스키마와 관련있는 데이타를 포함한다.

위의 단계 중에서 IRDS가 관리하는 단계는 (1), (2), (3)이며, 단계 (4)의 데이타는 DBMS 혹은 화일 시스템이 관리한다. 그림 2에서와 같이 인접한 두 단계 사이에는 하나 의 단계쌍이 존재한다. 단계 쌍의 상하위 단계 관계는 타입/인스탄스(type/instance) 관계지만 상위 단계는 하위 단계 데이타에 대한 제어 정보까지 포함한다. 그림 3은 ER(Entity－Relationship)모델에 대한 정보들을 IRDS를 이용해 각 단계별로 어떻게 나타내는가를 이해를 돕기 위해 본 논문에서 만든 예이다. 이는 모델링에 대한 스키마 정보가 IRDD 단계에서 정의되어 이용됨을 보여준다. 또한, IRD 단계쌍과 IRDD 단계쌍은 같은 데이타 모델링 기법(ER 모델)을 이용해 유지되지만, 응용 단계쌍은 하나 이상의 데이타 모델링 기법(관계형 모델, ER 모델, 객체 지향 모델등)을 지원할 수 있다.

## 3.2 IRDS 서버스 인터페이스의 개념

그림 4는 IRDS에서 사용되는 중요한 인터페이스들을 나타낸 것이다. 그림에서 각 프로세서는 해당 프로세서의 왼쪽에 있는 클라이언트(client)에 대한 서버(server)로서 작용한다.

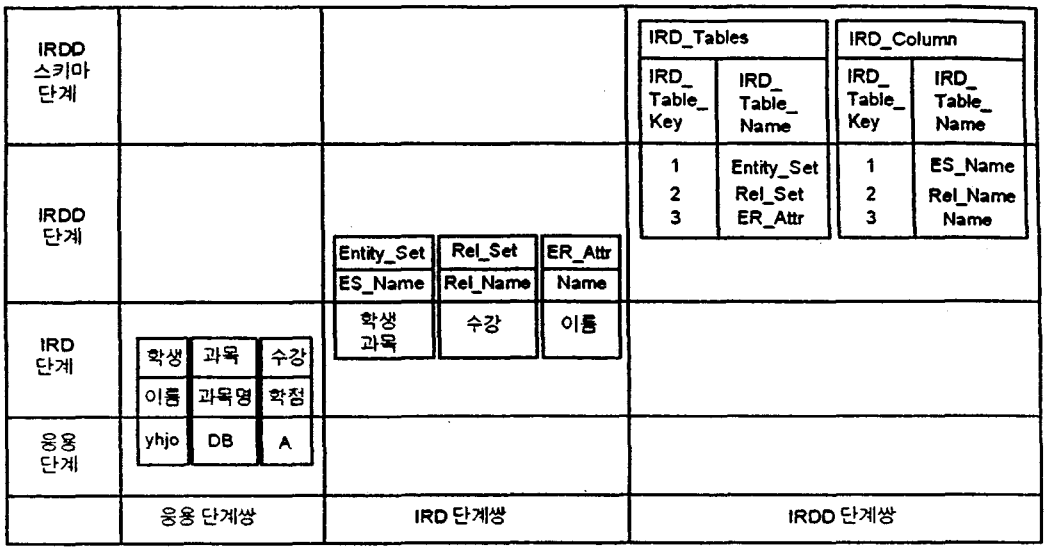
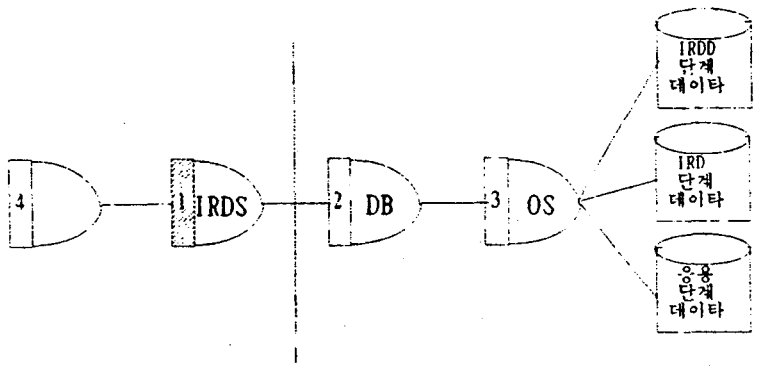



그림 3. ER 모델에 대한 IRDS의 정보자원 표현

Fig. 3 IRDS information resource representation of ER model




1 IRDS 서비스 인터페이스 2 데이타베이스 서비스 인터페이스

3 OS I/O 프로세서 인터페이스 4 기타 인터페이스들

그림 4. IRDS 프로세서 인터페이스들

Fig. 4 IRDS processor interfaces

그림에서 1 부분인 IRDS 서비스 인터페이스는 IRDS 데이타들을 접근하거나, 조작하는데 사용되는 프로세서간 인터페이스로 모든 필요한 서비스를 제공한다. 이것은 IRDD 단계 및 IRD 단계의 데이타들에 접근을 원하는 프로세서들에 의해 사용될 수 있으며, IRD 단계와 IRDD 단계의 모든 접근은 IRDS 서비스 인터페이스를 통해서만 이루어진다. 적당한 데이타 단계에 있는 데이타들을 접근하기 위해서 IRDS 서비스 인터페이스 프로세서는 데이타베

이스 서비스 인터페이스에서 제공되는 서비스들을 사용할 수 있으며, 또한 운영 체제 서비스들을 사용하거나, 직접적으로 데이타들을 사용할 수도 있다.

IRDS 서비스 인터페이스에서 제공되는 기능들은 데이타 모델링 기능(data modeling facilities), 버전 제어 기능(version control facilities), 이름 주기 기능(naming facilities), 한계치와 생략값에 대한 기능(facilities for limits and defaults), 다른 부가 가치 기능(other added value facilities) 및 life-cycle 기능등이 있다. 이들 기능은 IRDD 단계와 IRD 단계를 위한 테이블에 기록된 데이타들에 의해 제공된다.

# Ⅳ. IRDS 서비스 인터페이스 프로세서의 설계 및 구현

## 4.1 서비스 인터페이스들간의 상관관계 설계

그림 5의 IRDS 서비스 인터페이스들간의 관계는 표준[5]에 기술된 내용을 기반으로 본 논문에서 재정립한 것으로써, 여기에는 각 단계별 이용 가능 서비스 및 IRD 스키마 그룹, IRD 및 워킹셋(WorkingSet)의 상태 변환에 대한 정보 뿐 만 아니라, IRDS 서비스 인터페이스를 사용하기 위한 절차적인 흐름에 관한 것들을 기술하고 있다.

그림 5에서 "Irds"로 시작하는 접두어는 IRDS 서비스 인터페이스의 서비스를 나타내며, 사각형은 IRDS내에서 사용하는 객체 모임의 단위를 나타내고, 모서리가 둥근 사각형은 연산 모임의 단위를 나타낸다. 그림에서 워킹셋이라고 하는 것은 프로그램, 화면 설계, 데이타베이스 스키마, 또는 사업 모형과 같은 개념을 기술하기 위한 객체들을 단일 단위로 다루기 위해 사용자에 의해 정의된 것을 말한다.

그림에 나타나 있듯이, 먼저 IRDS 서비스 인터페이스에 응용을 연결하기 위해서는 Drop IRDD, Create IRDD와 Open IRDD 서비스를 이용해야만 한다. Drop IRDD 서비스는 시스템내에 존재하는 한 IRDD를 삭제해주는 서비스로서 IRDD를 삭제함에 따라 해당 IRDD가 제어하는 IRD 단계 및 응용 단계의 모든 데이타 및 스키마가 함께 삭제된다. IrdsOpen 서비스는 입력되는 매개 변수에 따라 IRDD 단계 또는 IRD 단계에 대한 서비스가 가능하도록 하나의 세션을 열어준다. Create IRDD 서비스는 시스템내에 새로운 IRDD를 생성해주고, 내부적으로 IrdsOpen 서비스를 실행하여 새로 생성된 IRDD에 세션을 열어준다. 물론, 각 응용이 IRD 서비스 인터페이스와의 연결을 끊기 위해서는 IrdsClose 서비스를 이용해야 한다.

IrdsOpen 서비스를 이용해 IRDS 서비스 인터페이스와 연결된 각 응용은 그림에 나타나 있듯이, IRDD 단계에 연결이 되었으면, ①에 해당하는 서비스들을 실행할 수 있고, IRD 단계에 연결이 되었으면, ②에 해당하는 서비스만을 실행할 수가 있다.

먼저 ②에 해당하는 서비스들을 보면, ②는 한 세션을 나타내며, 세션은 다시 트랜잭션들로 구성된다. 한 트랜잭션에서 수행할 수 있는 서비스들은 ③에 해당하는 서비스들과 Prepare, Commit 그리고 Rollback 서비스등이다.

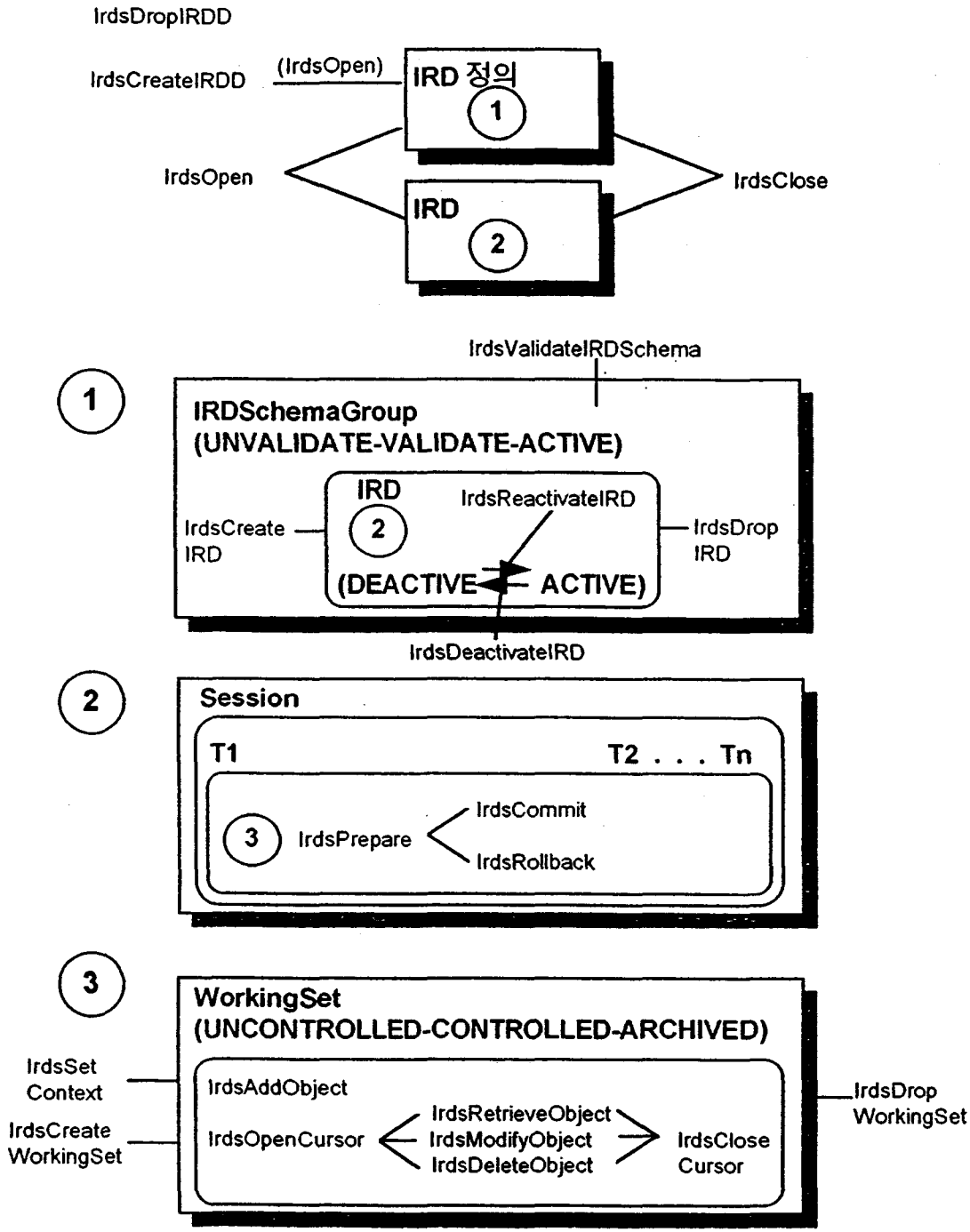



그림 5. IRDS 서비스들 사이의 관계성

Fig. 5 Relationship between IRDS services

먼저, Prepare 서비스는 ③에서 수행된, 즉 한 트랜잭션에서 수행된 모든 작업이 결함이 없음을 보장받기 위해 수행되는 서비스로서, two-phase commit 프로토콜에 따른다. 이 서비스가 수행된 후 각 Commit 서비스나 Roll-back 서비스를 이용하여 한 트랜잭션을 끝낼 수 있는데, 이들은 각각 한 트랜잭션내에서 이루어진 작업을 영구히 적용시키는가, 혹은 취소시킨다. 그밖에, 이들 서비스는 한 트랜잭션내에서 열었던 모든 커서들을 닫아준다.

③에 나타난 서비스들을 보면, ③은 한 문맥을 열어서 해당 문맥(context)내에서 일어날 수 있는 서비스들을 나타낸 것이다. 먼저 하나의 문맥을 열기 위해서는 Set Context서비스와 Create Working Set 서비스를 이용할 수 있는데, 이들은 각각 기존의 워킹 셋을 문맥화하거나, 새로운 워킹 셋을 만들어서 그 워킹 셋을 현 문맥으로 사용할 수 있음을 보인다. 문맥으로 사용된 워킹 셋은 life-cycle 제어를 위해 그 내용 상태가 바뀔 수 있는데, 이를 위한 것이 바로 Modify Content Status 서비스이다.

한 문맥내에서 가능한 서비스들은 현 문맥내에 새로운 객체를 더할 Add Object 서비스와 원하는 객체들에 커서를 열어주는 Open Cursor 서비스와, 이를 이용하여 객체들을 검색하기위한 Retrieve Object 서비스, 현 커서 위치의 객체를 삭제하는데 사용되는 Delete Object 서비스 그리고, 현 커서 위치의 객체를 수정하는 Modify Object 서비스 그리고, 열려진 커서를 닫는 Close Cursor 서비스들이 있다.

물론, 새로운 문맥을 선택하기 위해서는 다시 Set Context 서비스를 이용하면 되고, 그 밖에 기존의 워킹 셋을 삭제하기 위해서는 Drop Working Set 서비스가 이용될 수 있다.

①은 IRDD 단계에서만 가능한 서비스들(한 IRD 스키마 그룹과 그 그룹내 IRD에 대한 서비스들)과 위에서 설명한 ②에 관련된 서비스들로 구성되어 있다. 먼저, 현 문맥으로 사용되는 워킹 셋에 포함되는 한 IRD 스키마 그룹을 유효화 시키기 위해 이용되는 Validate IRD Schema Group 서비스가 이용될 수 있다. 그 밖에, IRD 스키마 그룹내의 IRD에 대한 연산들로는 새로운 IRD를 생성하기 위한 Create IRD, 기존의 IRD를 삭제하기 위한 Drop IRD, 기존의 IRD를 활성화 시키기 위한 Re-activate IRD, 기존의 IRD를 비활성화 시키기 위한 Deactivate IRD 서비스들이 있다.

## 4.2 서비스 인터페이스의 주요 지원 기능

### 4.2.1 데이타 모델링 기능

한 IRDD는 하나 이상의 IRD 스키마 그룹으로 구성된다. 여기서, 각 IRD 스키마 그룹이 각 데이타 모델링에 관련된 정보를 갖게 된다. 예를 들면, 관계형 모델에 관련된 정보와 객체지향 모델에 관련된 정보를 3.1절의 그림 3과 같이 입력하여 운영하게 되면, 사용자가 원하는 임의의 모델이라도 관리가 가능하게 된다.

3.1절의 그림 3에서 나타난 것과 같이, IRDD 스키마 단계의 데이타들은 모두 동일하지만, IRDD 단계및 그 아래 단계의 데이타들은 각 모델링 기법에 따라 다르게 유지된다. 이는 이들 각 모델링에 대한 스키마 정보가 IRDD 단계에서 테이블의 형태로 존재하며, 각 모델링을 이용해 구성한 응용의 스키마 정보는 IRD 단계에서 테이블의 형태로 존재하게 되기 때문이다.

이 기능과 관련된 서비스들은 IRDS내의 객체에 대한 연산들, 즉 Add Object 서비스, Retrieve Object 서비스, Modify Object 서비스, 그리고 Delete Object 서비스등이 있다.

#### 4.2.2 버전 제어 기능

버전 제어 기능은 IRDD 단계와 IRD 단계에서 동일하게 적용된다.

버전 제어 기능의 예를 들면, 아래 그림과 같이 각 부서별 워킹셋을 만든 후에, 각 부서의 모임에 대한 "H/W 사업부"와 "S/W 사업부" 워킹셋과 모든 사업부에 대한 워킹셋의 모임에 대해 "회사"라는 워킹셋을 구성할 수 있다. 또한, 또 다른 워킹셋의 관리로서, "영업부 V1.0" 워킹셋에 대한 새로운 버전인 "영업부 V2.0"을, "영업부 V2.0"에 대한 새로운 버전인 "영업부 V3.0" 및 "영업부 V2.1"을 생성 및 관리하고 있는 모습을 보인다.

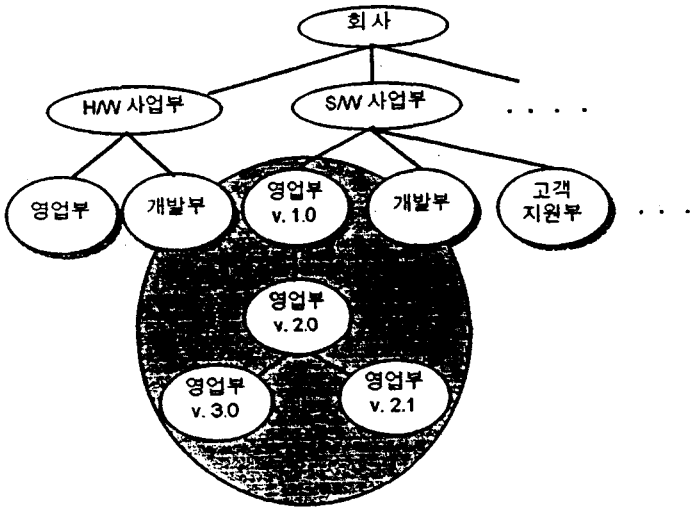



그림 6. 버전 제어 기능의 예

Fig. 6 An example of version control facility

따라서, 이와 같은 방법으로 워킹셋을 구성, 관리하게 되면, 각 정보에 대한 중복없는 공유를 유지할 수 있으며, 또한 워킹셋 구조에 의해 하위 단계인 응용에 대한 변화 과정에 대한 정보를 관리할 수 있게 된다. 중복없는 공유란 객체지향 모델의 특징인 상속성(inheritance)의 형태로 공통 정보를 공유할 수 있기 때문에 중복을 최소화 시킬 수 있는 장점을 가질 수 있으며, 또한 추가 변화 과정에 대한 정보를 관리할 수 있게 됨으로써 이전 버전으로의 복귀내지는 새로운 응용을 개발할 때 기존 프로그램 또는 정보를 재사용할 수 있다는 장점을 갖을 수 있다.

이 기능과 관련된 서비스들은 Create Working Set 서비스, Drop Working Set 서비스, Add Object Service, Modify Object 서비스, Retrieve Object 서비스, 그리고 Delete Object 서비스등이 있다.

## 4.3 서비스 인터페이스 프로세서의 구현

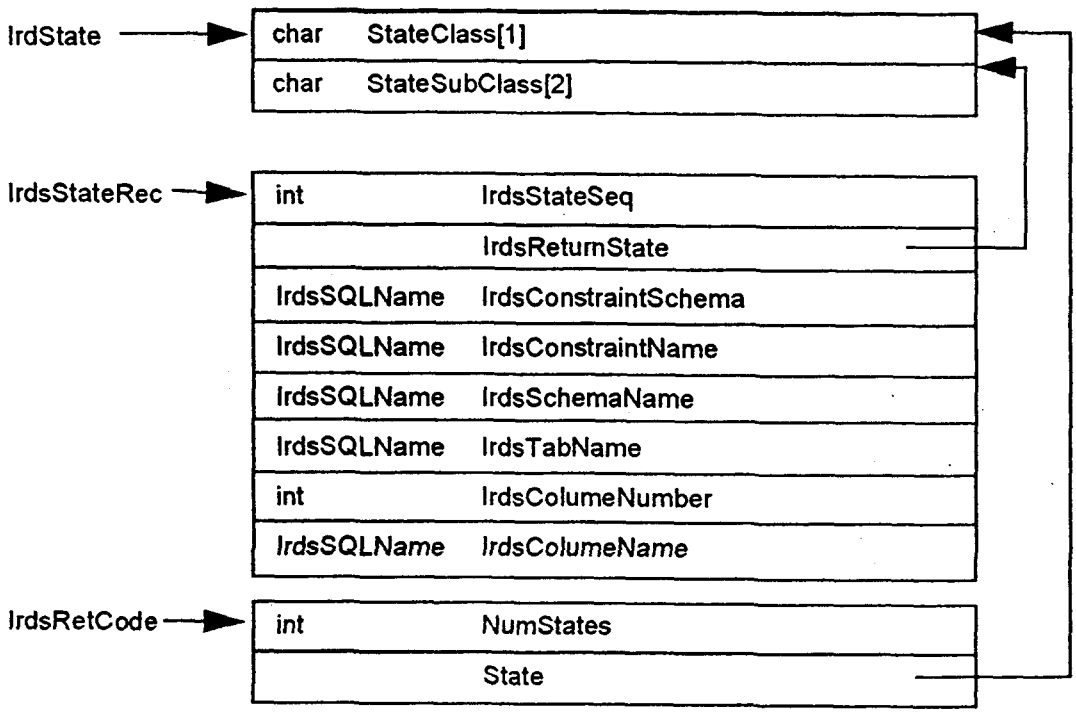
### 4.3.1 구현 개요 및 데이타 구조

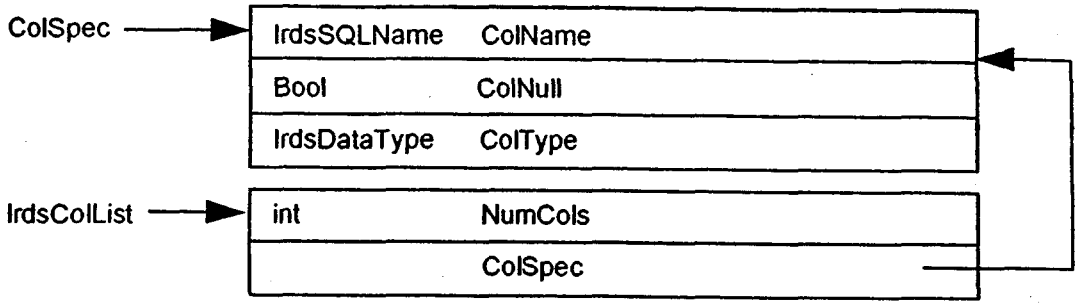
본 연구에서 구현한 IRDS 서비스 인터페이스 프로세서는 INFORMIX 관계 DBMS상에서 구현하였다[2]. 구현 언어로는 INFORMIX의 4GL을 이용하였으며, 구현 환경으로는 SUN Sparc Station을 이용하였다.

구현된 IRDS 서비스 인터페이스의 특징으로는 일상적인 정보를 저장하기 위한 스키마 정보를 IRD 단계가 관리하며, 마찬가지로 IRD 단계의 스키마를 IRDD 단계에서 관리함으로써 메타 데이타의 관리를 효율적으로 관리할 수 있다는 것이다. 그리고 워킹셋이라는 객체들의 모임을 포함하는 개념을 이용하여 버전을 관리할 수 있는것도 특징이며, 이러한 버전의 관리를 통해 버전들간에 중복되는 데이타를 유지할 필요가 없어졌다. 예를 들면, IRDD 단계과 IRD 단계 사이에 공통적으로 필요한 정보를 A라는 단일 워킹 셋에 모아 두고 A로 참조 경로를 지정하면 공통된 정보의 중복 관리를 피할 수 있다.

본 구현에서는 IRDD 단계, IRD 단계 및 응용 단계의 데이타를 3개의 DB에 독립적으로 나누어 구축하였다. 그리고, 그들이 상호 관계성을 갖도록 인터페이스(Interface)를 구현하였다. 이렇게 함으로써 각 단계 상호간 독립성을 유지하며, 효율적인 접근 방법을 모색할 수 있다.

IRDS 표준[5]에는 구현을 위한 명세가 되어있지 않다. 본 논문에서는 구현을 위해 중요한 데이타 구조를 C언어의 형태로 다음과 같이 정의하였다.






Get Diagnostics 서비스를 제외한 각 서비스를 끝낼 때, 서비스 인터페이스에 의해서 진단 영역(diagnostics area)에 하나 이상의 되돌려진 상태들이 기록된다. 이런 상태 레코드(state record)의 구조를 다음과 같이 C의 구조 형태로 정의하였다.

이때 IrdsRetCode 변수는 모든 서비스의 성공 여부를 나타내는 변수이다.

서비스에서 다른 서비스로 데이타 값들을 전달하기 위해 사용되는 행 리스트 매개 변수들의 구조는 다음과 같이 정의하였다. 여기서 IrdsDataType은 CHAR[N], DECIMAL[M,[N]], FLOAT, SMALLFLOAT, INTEGER, SMALLINT, MONEY, SERIAL, DATE등의 타입을 갖는다.

#### 4.3.2 각 서비스 함수의 구현

4.1절에서 설계한 각 서비스 인터페이스는 구현시에 C 함수 형태로 구현되었다. IRDS 표준[5]에는 구현과 관련된 구체적 언급이 없으며, 따라서 본 논문에서는 이를 다음과 같은 매개변수와 함수 형태로 구현하였으며, 전체는 크게 24개의 함수로 구현되었다. 여기서는 지면 관계상 IRDS의 IRD 단계에서 사용되는 주요 함수만을 예시하였다.

| 함수명 | 매개변수 | 설명 |
|---|---|---|
| void IrdsOpen | UserId IrdsUser; | 접근 권한을 조사하기 위한 사용자 이름 |
| | IrdsImpName IrdsDefName; | 접근하고자 하는 IRDD의 이름 |
| | IrdsImpNmae IrdsDicName; | 접근하고자 하는 IRD의 이름 |
| | Bool WillUpdate; | 각 단계의 접근시 정보의 수정 여부 |
| | IrdsSessId *CurrSessId; | 시스템에 의해 정의된 세션의 식별자 |
| | IrdsRetCode *RetCode; | 수행 결과 코드 |
| void IrdsGetDiagnostics | IrdsSessId CurrSessId; | 시스템에 의해 정의된 세션의 식별자 |
| | IrdsTrnsType CloseType; | 세션 종료시 트랜잭션 종료 타입(Commit/Rollback) |
| | IrdsRetCode *RetCode; | 수행 결과 코드 |

| | | |
|---|---|---|
| void<br>IrdsSetContext | IrdsSessId *CurrSessId;<br>IrdsName SessWkgSetName;<br>IrdsName SessWkgSetVerId;<br>Bool WillUpdate;<br>IrdsRetCode *RetCode; | 시스템에 의해 정의된 세션의 식별자<br>문맥으로 삼을 워킹셋 이름<br>문맥으로 삼을 워킹셋 버전 식별자<br>선택된 워킹셋에 대한 수정 여부<br>수행 결과 코드 |
| void<br>IrdsRetrieveObject | IrdsSessId CurrSessId;<br>IrdsCurId ObjCurId;<br>IrdsColList RequestedCols;<br><br>IrdsRetCode *RetCode; | 시스템에 의해 정의된 세션의 식별자<br>시스템에 의해 열려진 커서의 식별자<br>검색된 레코드의 정보를 저장하기 위한 리스트<br>수행 결과 코드 |
| void<br>IrdsModifyObject | IrdsSessId CurrSessId;<br>IrdsCurId ObjCurId;<br>IrdsColList ModifiedCols;<br><br>IrdsRetCode *RetCode; | 시스템에 의해 정의된 세션의 식별자<br>시스템에 의해 열려진 커서의 식별자<br>수정할 레코드의 정보를 저장하고 있는 리스트<br>수행 결과 코드 |
| void<br>IrdsCreateWorking Set | IrdsSessId CurrSessId;<br>IrdsName NewWkgSetName;<br>IrdsName NewWkgSetVerId;<br>IrdsName NewDcs;<br>IrdsName BasisWkgSetName;<br><br>IrdsName BasisWkgSetVerId;<br><br>IrdsRetCode *RetCode; | 시스템에 의해 정의된 세션의 식별자<br>생성하고자 하는 워킹셋 이름<br>생성하고자 하는 워킹셋 버전<br>생성될 워킹셋의 상태(Uncontrolled)<br>생성될 워킹셋의 기반이 되는 워킹셋 이름<br>생성될 워킹셋의 기반이 되는 워킹셋 버전 식별자<br>수행 결과 코드 |

# V. IRDS 서비스 인터페이스 프로세서의 응용

## 5.1 적용 대상 및 각 단계 스키마의 구축

본 논문에서 구현한 서비스 인터페이스 프로세서의 유용성을 보이기 위해 여기서는 어떤 조직체의 화일에 기반을 둔 응용 시스템(프로그램)들의 개발 및 관리의 예를 들어 IRDS의 기능중 특히 데이타 모델링 및 버전제어 기능을 어떻게 사용하는가를 설명하고자 한다.

IRDS에 필요한 모든 정보는 앞에서 언급한 것과 같이 IRDD단계, IRD단계, 응용 단계의 세가지 데이타베이스로 유지, 관리된다. 이중 IRDD의 스키마는 모든 응용에 독립적으로 공통되며, IRD의 스키마 응용에서 사용하는 데이타 모델에 따라 달라지며, 응용 단계 스키마는 응용 시스템에 따라 달라진다.

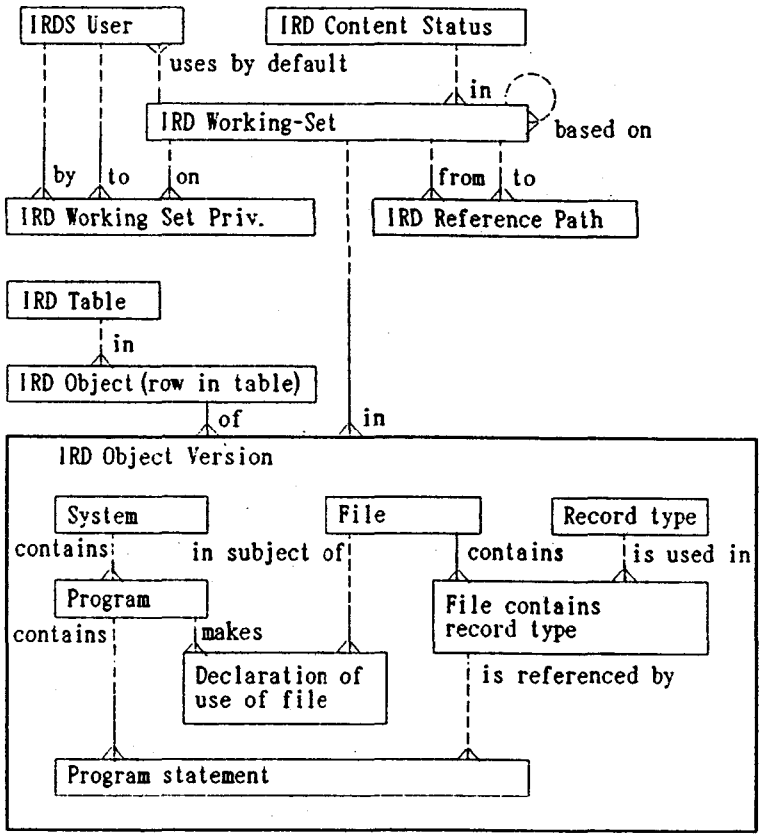



그림 7. 화일 시스템 기반 응용에 필요한 IRDS 테이블들

Fig. 7 IRDS tables for file-based application systems

그림 7은 본 논문에서 다루려고 하는 예에서 필요한 기본적 테이블 및 그 관계성을 도시하고 있다.

그림에서 IRDS User등은 IRDD 스키마의 테이블이며 그림에 나타나지 않은 것까지 포함하면 총 28개의 테이블이 있다. IRD Object Version속에 속하는 System, File등의 테이블은 IRD에 관련된 스키마이며, IRD는 IRDS의 적용대상이 사용하는 데이타 모델 요소들을 나타내므로 이에 따라 달라진다. 예를 들어 SQL 응용 개발을 위해서 필요한 IRD 스키마는 [8]의 표준에 정의되어 있다.

그림 7에서 선들은 각 테이블들 간의 관계성(relationship)을 나타낸 것으로 실선은 필수적(mondatory)관계, 점선은 선택적(optional)관계, 선끝의 삼각형은 1대 다(1-to-many)의 관계를 나타낸다. 예를 들어 그림7에서 System 테이블은 여러개의 program 객체를 포함한다는 관계성이 설정되어 있다.

본 연구에서는 IRDD와 IRD의 각 테이블을 생성하고, 각각 데이타베이스를 구축하였다. 예를 들어, IRDD단계의 IRD-WORKING-SET 테이블과 IRD단계의 FILE테이블은 다음과 같이 생성된다.

```
CREATE TABLE IRD_WORKING_SET
(
  WORKING_SET_KEY              IRDS_KEY,
  WORKING_SET_NAME             CHAR_DATA,
  WORKING_SET_VERSION_NAME     CHAR_DATA,
  BASED_ON_WORKING_SET_KEY     IRDS_KEY,
  IRD_CONTENT_STATUS_OBJ_KEY   IRDS_KEY,
  IRD_CONTENT_STATUS_WS_KEY    IRDS_KEY,
  OWNER                        IRDS_KEY,
  VERSIONABLE                  BOOLEAN NOT NULL
);

CREATE TABLE FILE
(
  FILE_KEY                     IRDS_KEY NOT NULL,
  FILE_NAME                    CHAR_DATA,
  NO_OF_REC_TYPES              SMALLINT,
  OWNER                        IRDS_KEY,
  CONTAINS_REC_TYPE            IRDS_KEY,
  FILE_TYPE                    CHAR_DATA,
  VERSIONABLE                  BOOLEAN NOT NULL
);
```

이 때 FILE 테이블은 CONTAINS_REC_TYPE 테이블을 통해 관련된 레코드 타입과 연관된다.

## 5.2 서비스 인터페이스의 응용 실험 및 평가

본 논문에서 구현된 서비스 인터페이스 프로세서의 각 함수(서비스)를 이용할 때 일일이 프로그램을 작성하여 사용해야 하는 번거로움을 줄이기 위해 본 논문에서는 메뉴 방식의 사용자 인터페이스를 구현하였다.

구현된 IRDS 서비스 인터페이스의 메뉴 계층도는 그림 8과 같다.

그림 8에서 ⑦표시에 해당하는 직사각형 모양은 IRD 단계가 아닌 경우, 즉 IRDD 단계의 접근을 원할 때, 이용 가능한 서비스들을 나타내는 것으로 IRD 단계에서 제공되는 서비스를 포함함을 나타낸다.

다음은 구현된 IRDS 서비스 인터페이스를 통해 IRDS의 효용성을 실험한 결과이다.

### 5.2.1 버전제어 기능

본 논문에서 예로 사용한 화일에 기반을 둔 프로그램 개발 및 관리부분에서 응용 환경의 변화, 즉 어떤 회사의 성장이나 경영 정책의 변화등에 의해 프로그램을 변경하려면 이에 대한 버전 관리가 매우 중요한다. 이를 위해서는 어떤 CASE 도구나 형상 관리(configuration management) 도구를 사용해야 하며, 특히 이들 도구는 프로그램 수정 위주로 버전 관리를 행하므로 이 프로그램과 관련된 화일등의 변화가 일어났을 때 이 화일을 공유하는 여러 프로그램이 존재한다면 이들 프로그램을 일일이 찾아내어 수정해야 하는 어려움이 있다.

하지만, IRDS를 이용하면 적용 대상의 각 정보를 통합적으로 관리하므로 변경 부분에 관계되어 있는 모든 부분을 일관성있게 수정하여 버전 관리가 가능해진다.

다음은 그림 6에 나타난 영업부 관리 프로그램의 버전 관리의 한 예를 보여준다. 현재 영업부 관리 프로그램 및 이에 관련된 화일들이 있다고 할 때, 필요에 의해 영업부 관련 정보의 화일 형태를 인덱스된 순차화일(indexed sequential file)로 바꾸었다고 하자. 이 때 IRDS의 각 서비스를 이용하여 버전관리를 하

는 방법은 다음과 같다.

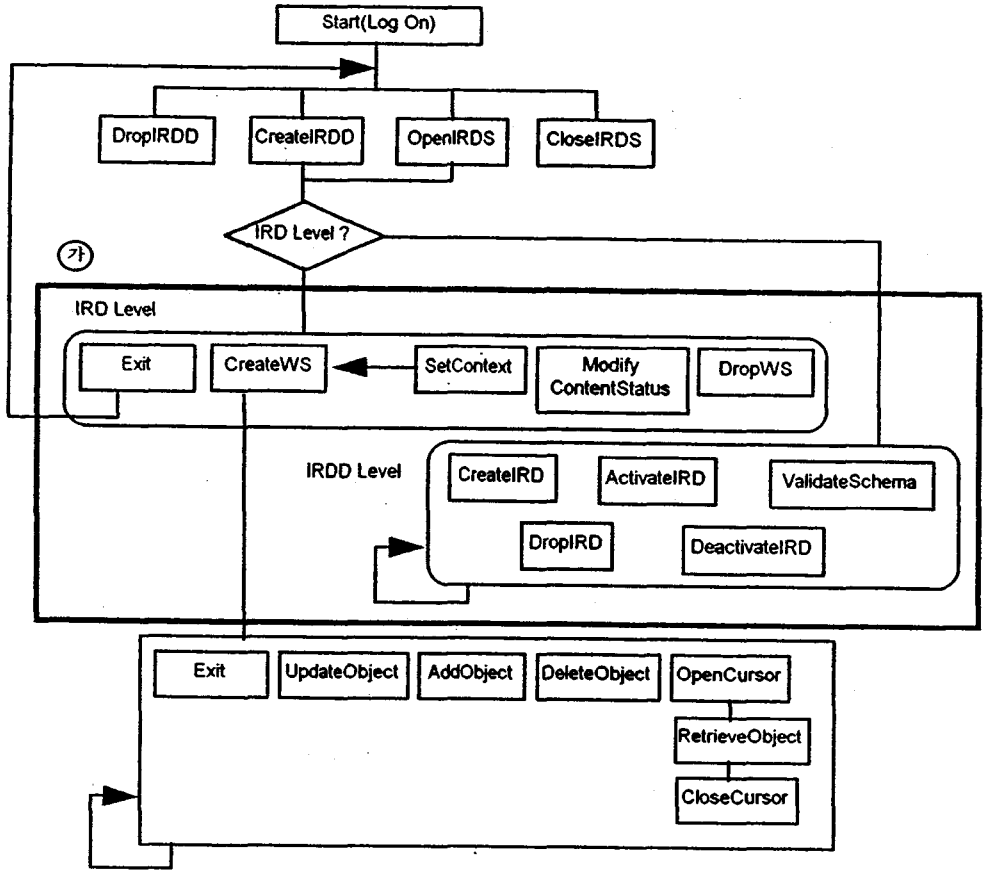



그림 8. IRDS 서비스 인터페이스의 메뉴 계층

Fig 8. Menu hierarchy of IRDS Services Interface

먼저 OpenIRDS 서비스를 이용하여 유용한 사용자인지를 검사한 후, 필요한 IRDD와 IRD를 연다.

〈화면 1〉에서 사용자는 IRD 단계에 워킹셋을 생성하기를 원하기 때문에 1을 선택하며, 1이 선택되면, IRDS는 디폴트 워킹셋의 사용여부를 묻게 된다. 이 때, 사용자가 N를 입력하면, IRDS서비스 인터페이스는 문맥으로 사용할 워킹셋을 묻게되고, 사용자는 워킹셋 "영업

LEVEL SELECTION MENU

1. IRD LEVEL
2. IRDD LEVEL
3. Return to TOP MENU

Type the number or letter of the option that you want, or a question mask (?) for help >> 1

〈화면 1〉

부"를 입력하면, IRDS는 사용자가 해당 워킹셋에 대한 사용 권한 여부를 조사한 후, 사용 권한이 있게 되면 다음 화면을 보여준다.

```
                    IRD LEVEL

1. Drop Working Set      2. Create Working Set
3. Set Context           4. Modify Content Status
5. Add Object            6. Delete Object
7. Modify Object         8. Retrieve Object

          e. Return to LEVEL SELECTION MENU

Type the number or letter of the option that you
want, or a question mask (?) for help >> 2
```

〈화면 2〉

〈화면 2〉는 IRD단계에서 수행할 수 있는 서비스들을 나타낸 것이다. 이 때, 사용자는 새로운 워킹셋인 영업부 V2.0을 생성하기 위해 2번을 선택한다.

```
Working Set Table

WS Key [00020001]
WS Name [영업부 ]  SQL WS Version Name [V2.0 ]
WS Owner Key  [00040009]
Versionable    [Y]
Based WS Key     [00020000]
```

〈화면 3〉

〈화면 3〉은 새로운 워킹셋을 입력받기 위한 화면으로서, 사용자는 영업부 V1.0의 새로운 버전인 영업부 V2.0을 입력하며, 자동적으로 Based WS Key 필드에는 현재 문맥으로 선택된 영업부 V1.0의 키값이 표시된다. 입력이 끝나면, 화면은 다시 〈화면 2〉로 돌아간다. 이 때, 사용자는 3을 입력하여, 현 문맥을 영업부 V2.0으로 바꾼 후, 화일 형태를 수정하기 위해 7번을 입력하게 된다. 7번을 입력하게 되면, IRDS 서비스 인터페이스는 워킹셋 영업부 V2.0이 갖는 모든 테이블들을 〈화면 4〉와 같이 표시해 준다.

```
1. System                2. Program
3. File                  4. Declaration of use of file
5. Record type           6. File contains record type
7. Program statement

Type the number or letter of the option that
you want, or a question mask (?) for help >> 3
```

〈화면 4〉

〈화면 4〉에서 사용자는 수정을 원하는 객체, 즉 사번 객체를 갖는 테이블인 File 테이블인 3을 입력한 후, 다음 화면과 같이 수정을 한다.

```
File

File Key [10000102]          File Name [영업부ISAM]
No. of Rec. Types  [ 1 ]     Owner [00040009]
Contains Rec. Types [7007011] File Type [CISAM]
Versionable   [ Y ]
```

〈화면 5〉

〈화면 5〉에서 새로 변경된 ISAM 형식의 화일에 대한 정보를 입력하면, IRDS 사용자 인터페이스는 이에 관련되어 바꾸어져야할 레코드 타입의 정보 및 프로그램, 헤더 화일들 심지어는 관련된 프로그램 문장까지도 반복하여 자동적으로 사용자에게 변경 화면을 제시하여 관련된 정보의 일관성 있는 갱신을 행하게 된다. 또한 사용자가 만일 전의 버전을 보기 원하면 워킹셋을 바꾸어 전의 문맥으로 돌아갈 수도 있다.

한가지 여기서 오해의 소지가 있는 사항은 File Key등의 정보를 일일이 사용자가 명시하는 것처럼 보이나, 이는 단지 이해를 돕기 위해 사용자 인터페이스상에 보이는 것이며 실제 IRDS 자체 내에서 생성하고 관리하는 값임을 밝혀둔다.

위와 같이 IRDS의 버전 기능을 이용하면 관련된 정보중 어느 일부분이 수정되어도 이에 영향받는 모든 부분의 갱신이 일어나 일관성 있는 버전 관리가 되며, 만일 여기에 관련된 H/W정보나 스프레드 쉬트(spread sheet)등 다른 도구의 정보도 추가한다면 이들 정보 자원들을 통합 관리하는 효과를 보게 되는 잇점이 있다.

### 5.2.2 데이타 모델링 기능

위의 예에서 다루는 화일들이 만일 다른 플랫폼(flatform), 즉 PC 또는 워크스테이션, IBM 대형 컴퓨터등에 존재하는 것들이라면 이 화일들에 대한 메타 데이타인 디렉토리 정보는 각 기계의 운영체제에 따라 각기 다른 방식으로 유지된다. 이를 통합하여 테이블 형태의 데이타 모델로 기술할 수 있도록 도와주는 것이 IRDS이다.

또한, 위의 예에 관련된 H/W 정보나 각 프로그램에 관련된 DTP로 작성된 문서(document)들이 있다면 이들 정보도 IRDS내에 동일한 형태로 통합 표현, 관리될 수 있다.

따라서 다른 데이타 모델을 따르거나 데이타 모델링 기능이 없는 정보들에 대해서도 IRDS를 통해 이를 일관되게 같이 유지할 수 있으며, 또한 IRDS 서비스를 통해 이들 정보를 검색, 수정, 조작할 수 있는 추가적인 기능을 제공받을 수 있다는 것이 바로 IRDS의 장점이다.

## VI. 결 론

본 연구에서는 IRDS 서비스 인터페이스 프로세서를 설계하고 구현하기 위하여 먼저 IRDS의 필요성과 유사한 시스템과의 차이점, IRDS의 기본 구조와 IRDS 서비스 인터페이스 프로세서에 대해 살펴 보았으며, 이들을 기반으로 데이타 구조와 각 서비스들을 재설계하였고 이를 INFORMIX 관계 DBMS상에 INFORMIX 4GL을 사용하여 구현하였다.

구현된 IRDS 서비스 인터페이스 프로세서를 화일에 기반한 응용 및 개발 부분에 적용하여 실험한 결과 다음과 같은 효용성을 확인할 수 있었다.

1) 다른 플랫폼, 다른 데이타 모델을 사용하는 응용 시스템의 정보 자원들을 하나의 형태로 통합.표현, 관리할 수 있었으며,
2) 기존의 관계 DBMS에서 제어하지 못했던 객체의 버전을 IRDS의 워킹 셋을 이용해 일관성있게 관리할 수 있게 되었으며,

3) 워킹 셋의 상속성을 통해 중복된 데이타를 최소화 시킬 수 있었다.

반면, 본 연구를 통해 드러난 문제점으로는, 구현을 위해 사용한 INFORMIX 버전이 1989년에 제정된 SQL1 언어를 지원하기 때문에 1992년에 제정된 SQL2에 기반으로 만들어진 IRDS 서비스 인터페이스 표준을 만족할 만큼의 기능을 갖지 못해 구현시 프로그램 내에서 부족한 기능을 직접 구현해야 하는 어려움이 있었다. 물론 이 문제점은 추후 관계 DBMS의 기능향상으로 해결되리라 기대한다.

향후 연구 계획으로는 구현된 서비스 인터페이스의 세계 표준이 계속 변경되고 있어 표준이 확정되는 부분에 대해서 계속적인 보강과 함께 구현을 해나가야 할 것이며, 이에 더하여 CASE나 형상 관리등 좀 더 실용적인 응용에 대한 적용 실험이 수행되야 할 것이다.

## 참고문헌


Hales, K. and Guilfoyle, C., "*The Future of the Database,*" Ovum Ltd., 1989.

INFORMIX Software, Inc., "*INFORMIX-ESQL/C*", Feb., 1986.

ISO/IEC JTC1/SC21/WG3 N1011, "*IRDS Rapporteur's Report to WG3 Florence Meeting Closing Plenary,*" 10th, Nov., 1989.

ISO/IEC IS10027, "*Information Resource Dictionary System(IRDS) Framework,*", 19th, Jan., 1990.

ISO/IEC JTC1/SC21/WG3 DIS 10728, "*Information Technology - Information Resource Dictionary System(IRDS) Services Interface,*" July, 1991.

Olle, T. W., "*ISO/IEC JTC1 SC21/WG3 Database Standards Work,*" in Proc. of the International Seminar for OSI Computer Communication, Seoul, 1990.

Prabandham, M., et. al., "*A View of the IRDS Reference Model,*" Database Programming and Design, Mar., 1990.

ISO/IEC JTC1/SC21/WG3 Database, "*IRDS:Design Support for SQL Applications*", Mar.,1990.

Prabandham, M., et. al., "*The Role of the IRDS,*" Database Programming and Design, Apr.,1990.

조 완섭, 김 명준, "*Information Resource Dictionary System,*" 정보 통신 기술, Vol.2, No. 3, Nov., 1988.