분산객체 환경에서의 IRDS 기반 정보저장소 설계 및 구현

염 태 진[†] · 박 재 형[††] · 리 자[†††] · 김 기 봉[††††] · 진 성 일[†††††]

요 약

정보저장소는 데이터베이스를 이용한 일차적인 정보이외에도 전 조직체의 정보 자원과 같은 복잡하고 다양한 정보의 관리를 위한 시스템이다. 정보저장소의 구현을 위한 국제 표준으로 현재 IRDS가 있으나 이는 분산 환경을 제공하지 않는다. 따라서 본 논문에서는 분산 환경하에서 운영 가능한 IRDS 기반의 정보저장소를 설계 및 구현한다. 이를 위해 객체 지향 개념을 제공하기 위한 정보저장소로서 ISO IRDS를 이용하며, 분산 객체 기술을 지원하기 위해 CORBA를 채택한다. 구현은 IRDS 표준에서 제시하고 있는 핵심적인 기능을 바탕으로 메타데이터 관리 테이블과 서비스 인터페이스를 구현하고, 이의 원활한 활용을 위해 윈도우즈 기반 사용자 인터페이스를 구현한다. 또한 본 논문에서 구현한 시스템을 활용하기 위해 다양한 응용 분야 중 하나인 워크플로우 관리 시스템에 적용하여 그 활용도를 검증한다.

A Design and Implementation of IRDS-based Repository on Distributed Object Environment

Tai-Jin Yeom[†] · Jae-Hyoung Park[††] · Li Zi[†††] · Ki-Bong Kim[††††] · Seong-Il Jin[†††††]

ABSTRACT

A repository is a system for management not only simple information using database but also complex and various information resources. A standard for implementation of a repository is now IRDS, but it does not support distributed-object environment. Therefore, in this paper, we design and implement IRDS-based repository on distributed-object environment. For implementation, we use ISO IRDS as a repository supporting object-oriented concepts, and choice the CORBA to support the distributed-object technology. Implemented contents are metadata table and service interface based on core functionality to be proposes at IRDS, and window-based user interface to use them. Also, we apply it to the workflow management system and test its practical usefulness.

1. 서 론

최근 정보처리가 급속히 발전함에 따라 조직체에서 사용하고 있는 방대한 양의 자료를 효율적으로 저장하고 관리하기 위한 데이터베이스 관리 시스템(DBMS)의 이용이 크게 증가하고 있다. 그러나, DBMS의 기능은 일차적인 단순한 자료 관리의 수준에 그치고 있으며, 전 조직체의 정보 관리 등의 복잡하고 다양한 정보 관리에 미치지 못하고 있다[14,15,18].

이러한 정보 처리 환경에서 보다 효율적인 정보자원

† 준 회 원 : 충남대학교 소프트웨어연구센터 전임연구원
†† 준 회 원 : 충남대학교 대학원 컴퓨터과학과
††† 비 회 원 : 충남대학교 대학원 컴퓨터과학과
†††† 정 회 원 : 대전보건대학 전산정보처리과 교수
††††† 종신회원 : 충남대학교 컴퓨터과학과 교수

논문접수 : 1998년 10월 1일, 심사완료 : 1999년 3월 23일

의 관리를 위한 개념이 도입되었고, 각 응용 시스템들이 요구하고 사용하고 있는 모든 정보자원을 통합, 저장, 관리할 수 있는 시스템이 생겨나게 되었는데 이것이 정보저장소(Repository)다[14,15].

정보저장소는 소프트웨어 생명 주기 동안 모아진 시스템 정보를 저장 및 관리하는 시스템으로 각 도구들, 각 개발 단계들, 각 사용자들, 각 프로세스들 사이의 시스템 정보를 유지한다. 정보저장소에서 관리되는 시스템 정보란 자료의 사용이나 유지를 기술하는 스키마 정보, 제어나 전 조직체의 정보 관리를 위한 경영 정책, 인적 자원, 업무 규칙, 조직 규칙, 정보 구조, 하드웨어/소프트웨어 환경 정보 등을 말한다. 이러한 정보저장소는 시스템 유지 보수를 용이하게 하고, 소프트웨어 재사용을 가능하게 하는 등의 소프트웨어 개발, 유지, 보수 등에서 비용의 감소라는 큰 이점을 내포하고 있다[7]. 또한 CALS/EC 구현의 요소 기술 중 하나로서 분산 환경에서 여러 시스템들과 통합하여 사용할 경우 보다 상위의 차원에서 각 시스템들이 가지는 메타데이터들을 총체적으로 관리함으로써 통합정보시스템 구축을 용이하게 한다[17].

이러한 정보자원 관리 개념을 가진 시스템이 많이 존재하는데[14,17] 그 중의 하나가 바로 정보저장소 표준인 IRDS(Information Resource Dictionary System)이다. IRDS는 조직체 내의 각 응용 시스템들이 사용하거나 필요로 하는 모든 정보자원을 시스템간에 공유하여 효율적으로 저장 및 관리하는 특별한 응용시스템이다[18]. 현재 IRDS 표준은 ANSI와 ISO라는 두 가지의 방향으로 연구가 진행되고 있다. 두 표준 모두 정보자원의 관리를 위해 단계를 두어 계층적으로 관리하며 모델링은 E-R 모델링을 근간으로 하고 있다. 그러나 ISO IRDS는 최근의 추세에 맞추어 객체 지향 모델링을 기반으로 하는 표준 작업이 진행 중이며, 또한 ANSI IRDS보다 제약 조건의 명시 및 기능 면에서 더욱 뛰어나다.

본 논문에서 구현하고자 하는 정보저장소는 ISO IRDS를 근간으로 하고 있다. 그러나 ISO IRDS 표준은 중앙 집중적 방식으로 다중 사용자들이 사용하는 정보저장소이며, 분산객체 클라이언트/서버 환경에서의 분산 기술과 객체 기술이 통합된 방식이 요구되고 있다. 분산 객체 환경에서의 정보저장소는 지역적으로 분산되어 있는 정보자원을 관리할 수 있으며. 독립적으로 개발된 소프트웨어 구성 요소들을 통합할 수 있는 프레임워크를 제공할 수 있으며, 기존의 정보저장소 기능을 분산 환경에서 동적으로 이용할 수 있다. 현재 분산객체 환경에서의 정보저장소로는 OMG 그룹의 표준안이 있으나 상용화된 제품은 거의 없는 실정이며 또한 ISO IRDS도 분산 객체 환경을 지원하기 위해 구현된 제품이 없다.

따라서 본 논문에서는 이러한 최근의 추세와 요구사항에 맞추어 분산 객체 환경하에서 정보자원을 효율적으로 관리하기 위해 표준화된 IRDS를 기반으로 정보저장소를 설계 및 구현하였으며, 이를 워크플로우라는 시스템에 적용하여 그 활용도를 입증하였다.

본 논문의 구성은 다음과 같다. 2장에서는 연구 배경으로써 정보저장소 기술과 분산 객체 기술, 워크플로우 관리 시스템에 대해 언급하고, 3장과 4장에서는 각각 분산 객체 환경하에서의 IRDS 기반 정보저장소의 설계 및 구현을 기술하며, 5장에서는 구현된 정보저장소를 워크플로우 관리 시스템에 적용하여 그 결과를 분석해 본다. 마지막으로 6장에서는 결론 및 향후 발전 방향에 대해 언급한다.

2. 배 경

분산 객체 환경하에서 정보자원을 효율적으로 관리하기 위해서는 분산 객체 관리 기술과 정보저장소 기술이 필요하며, 이의 검증을 위해 워크플로우를 적용하고자 한다.

2.1 분산 객체 환경

분산 객체 환경이란 생산성 있는 각 응용 프로그램을 작성하여, 파일이나 데이터베이스와 같은 저장 매체를 통해 분산 저장하여 정보를 공유하고, 서로 다른 운영체제와 네트워크 환경 등의 이종의 환경에서 작동 가능한 클라이언트/서버 환경을 말한다[10,12]. 이는 개발된 소프트웨어가 분산 환경에서 서로 다른 시스템들과 쉽게 통합될 수 있는 환경이라 할 수 있다. 그러나 이러한 분산 환경을 구축하기 위해서는 기존의 C와 같이 논리(logic)의 흐름에 따라 프로그램을 작성하는 방식으로는 이와 같은 요구사항을 해결하기 어렵고, 또 이종의 분산 환경에서 다양한 형태의 응용 프로그램을 통합하기 위해서는 일정한 결합 방법이 필요하다.

이런 문제를 해결하기 위한 표준으로는 RPC(Remote Procedure Call)나 OLE(Object Linking and Embedding),

COM(Common Object Model)/DCOM(Distributed COM) 등의 방식이 있으며, 최근에는 CORBA(Common Object Request Broker Architecture)를 이용한 분산 객체 환경이 주목을 받고 있다. RPC의 경우 복잡한 프로그램을 작성하는데 있어 많은 한계가 있고, OLE와 COM/DCOM 같은 경우는 윈도우 객체만의 호환을 지원하기 때문에 진정한 분산 시스템의 표준이라 말하기 어렵다[10,12]. 따라서 본 논문에서는 분산 객체 환경을 지원하기 위하여 CORBA를 기반으로 하고자 한다.

CORBA는 OMG(Object Management Group)에서 1990년에 제안한 표준으로 이기종간의 상이한 분산 환경에서의 상호 동작성을 제공하고 이질 객체 시스템간의 완전한 상호 연결을 위하여 제안된 것으로 객체간에 서비스를 요청하고 그것에 대한 응답의 투명한 메커니즘을 제공한다. CORBA의 기본 구조는 클라이언트와 서버, ORB(Object Request Broker)로 구성되어 있다. 클라이언트는 동적 호출 API를 사용하여 컴파일된 IDL 스텁(stub)을 거치거나 처리 중에 서비스를 정적으로 불러낸다. 서버는 정적 골격(skeleton)을 사용하여 클라이언트 요구를 객체 수행 프로그램에 연결하거나, 새로운 동적 골격 인터페이스를 사용하여 런타임으로 보내기도 한다[10].

2.2 정보저장소

정보저장소는 소프트웨어 개발의 각 단계에서 사용되고 생산되어지는 모든 정보와 시스템 요소들을 저장함으로써 계획에서부터 유지 보수에 이르기까지 소프트웨어 생명주기 전체를 지원한다. 이는 조직, 조직의 구조, 기업 모델, 프로시져, 데이터 모델, 데이터 객체, 객체의 관련성, 프로세스 모델, 메타 모델, 실질적인 코드, 프로젝트 관리 정보, 각종 문서 등에 관한 정보를 저장하는 지식베이스로 사용된다. 또한 정보저장소는 이러한 많은 형태의 정보뿐만 아니라 이러한 정보들의 상호관계와 이들 정보들의 유효성을 증명하고 사용하는 규칙도 포함하여, 시스템의 이해와 변화 관리를 지원한다[8,14]. 정보저장소의 응용분야로서는 감사 관리, 컴퓨터 응용, 형상 관리, 자료 관리, 데이터베이스 관리, 통신망 관리, 프로젝트 관리, 시스템 공학, 워크플로우 관리(Workflow Management) 등에서 적용되어 사용될 수 있다.

정보저장소 구현을 위한 표준으로는 CDIF(CASE Data Interchange Format), PCTE(Portable Common Tool Environment), ATIS(A Tools Integration Standard), SDAI(Standard Data Access Interface), IRDS (Information Resource Dictionary System) 등이 있다[14,17]. 그러나 이러한 표준들은 정보의 작성 규칙이나 처리 규정 등이 독립적이며 일정하게 통일된 양식을 가지고 있지 않다. 그 중 국제 표준으로 지정되어 연구되고 있는 정보저장소가 바로 IRDS이다. IRDS의 연구는 ANSI와 ISO에서 담당하고 있는데, 본 논문에서는 ISO IRDS를 이용하고자 한다.

ISO IRDS는 IRDS 기본 기능, IRDS 모델링 규칙, IRDS 테이블, IRDS 추상 자료 구조 및 서비스 인터페이스 등의 IRDS에 대한 개념과 구현 방안을 위한 표준안을 제시하고 있다. IRDS 테이블은 ANSI 표준과 마찬가지로 IRD와 IRD 정의라는 두 단계로 계층적으로 정의하며 두 단계 쌍들 간의 최대한의 병렬성을 유지하면서 테이블의 내용에 따라 네 가지 유형으로 분리하고 있다. 네 단계 유형이란 객체 관리를 위한 내부 테이블, 시스템 환경 관리를 위한 환경 테이블, 메타데이터 조작을 위한 한정 테이블, 공통 기능을 담고 있는 공통 테이블로서 이 중 메타데이터를 직접적으로 관리하는 한정 테이블이 두 단계의 쌍으로 구성된다. IRDS 추상 자료 구조는 SQL에 기반해 각 추상 자료 구조의 기능, SQL 정의, 각 구조의 참조 관계 등이 정의되어 있다. IRDS 서비스 인터페이스는 서비스로 제공되는 모든 함수를 세 개의 그룹 즉, 운영 서비스, 단계 독립 서비스, 단계 특정 서비스로 나누어 정의하며 각 서비스에 대한 기능, 형식, 입력, 출력, 일반규칙, 추상 자료구조에 대한 연산 등을 정의한다.

그러나 IRDS를 포함한 현재까지의 정보저장소는 중앙 집중적 방식으로 분산 환경에서 여러 시스템들과 연계하여 사용하기가 어렵다. 이러한 문제점들을 해결하기 위한 표준으로는 OMG에서 제안하는 정보저장소 표준이 있다. OMG 표준은 CORBA 중심의 분산 환경에서 관리되는 정보를 IRDS와 비슷하게 객체 모델(Repository Object Model), 서비스 모델(Repository Service Model), 내용 모델(Repository Content Model)로 나누어 계층적으로 관리한다. OMG 표준은 Unisys와 IBM 등에서 제품화를 위한 준비를 하고 있으나[14,17] Unisys의 Universal Repository나 IBM의 Teamwork은 CORBA와 같은 분산 객체 환경이 아니라 단순 RPC (Remote Procedure Call) 방식으로 작동하며, 저장되는

데이터들도 단계적으로 구분하여 관리되지 않고 있다.

따라서 분산 객체 환경하에서 여러 시스템들과 연계하여 사용할 수 있도록 하며, 저장되는 데이터가 잘 정의된 테이블에 단계적으로 관리될 수 있는 정보저장소가 필요하다.

2.3 워크플로우

워크플로우란 전반적인 비즈니스 목표를 성취하기 위한 규칙의 집합을 규정함으로써 워크플로우의 참여자 사이의 전달 되는 문서, 정보, 태스크 진행의 자동화를 말하며, 워크플로우 관리 시스템이란 다양한 작업 활동의 순서와 활동의 순서에 관련된 사람 사이의 정보 기술 자원의 호출을 관리함으로써 업무 프로세스의 자동화를 제공하는 시스템을 말한다[1].

이러한 워크플로우 시스템은 여러 소프트웨어 벤더와 대학 등에서 상용 제품으로 또는 연구용으로 많이 개발되고 있다[1,9,11]. 그러나 각각의 워크플로우가 제각각이며 표준화가 진행되지 않고 있다. 즉, 서로 공동으로 작업을 할 수 있도록 하는 표준의 정의가 필요하다. 이러한 이유로 만들어진 단체가 WfMC(Workflow Management Coalition)이다. 따라서 본 논문에서 기술하는 워크플로우 참조 모델은 WfMC의 참조 모델을 근간으로 하고 있다[1].

WfMC에서 제시하고 있는 워크플로우를 위한 일반적인 구조는 워크플로우를 해결하는 워크플로우 엔진, 프로세스 정의를 내리는 프로세스 정의 도구, 데이터를 저장하는 데이터베이스, 그리고 기타 사용자 인터페이스나 사용자 응용프로그램 등으로 구성되어 있다. 워크플로우 엔진은 프로세스를 해석하고, 프로세스들을 제어하며, 활동(activity)들의 순서를 정의하고, 정의된 활동들을 사용자 업무 요소(work item)에 추가시켜주며, 필요한 응용프로그램을 가동시켜준다. 엔진은 다양한 프로세스의 실행을 관리하는 하나 또는 그 이상의 워크플로우 엔진이 될 수 있다. 프로세스 정의 도구는 워크플로우 시스템이 사용하고 접근할 수 있는 프로세스 형태를 만드는데 사용된다. 사용자 인터페이스는 사용자와의 의사 소통과 제어를 담당하고, 사용자 응용프로그램은 워크플로우의 진행 중 필요한 메일 매니저나 워드 등과 같은 프로그램이다[1].

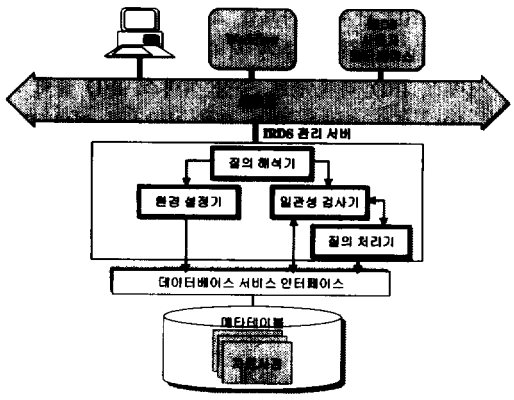
워크플로우는 여러 가지 데이터들을 사용한다. 이러한 데이터로는 프로세스 정의(Process Definition) 정보, 조직체(Organization)와 규칙(Role) 정보, 워크플로우 제어 데이터(Workflow Control Data), 워크플로우 관련 데이터(Workflow Relevant Data), 작업리스트(Work List) 등이다[9,11]. 이러한 데이터들을 관리하기 위해 기존의 워크플로우 관리 시스템들은 파일이나 단순 데이터베이스를 이용하였다. 그러나 파일이나 데이터베이스로는 워크플로우 운영 정책, 워크플로우 절차, 계획 등의 복합적인 정보자원을 관리 하기는 미흡하며, 단지 데이터베이스에 저장되는 응용 프로그램을 위한 단순 데이터의 관리 차원에 국한되어 있다. 따라서 워크플로우 관리 시스템에서 정보저장소를 사용하면 정보자원의 가용성을 높이며, 다양한 정보의 형태를 저장할 수 있으며, 다른 정보와의 연관성 등을 파악할 수 있도록 하며, 워크플로우의 데이터들을 정보저장소 내에 단계별, 계층적인 방법으로 효과적으로 관리를 할 수 있다.

3. 분산객체 환경에서의 IRDS 기반 정보저장소 설계

ISO IRDS 표준안에는 정보의 자료 참조, 메타데이터/메타스키마 조작, 버전제어, 보안을 비롯한 기타 부가적인 기능이 많이 포함되어 있으나, 본 논문에서 설계범위는 자료참조 및 메타데이터/메타스키마 조작에 필요한 핵심이 되는 서비스 인터페이스와 이의 원활한 사용을 위한 사용자 인터페이스 만을 설계하고자 한다. 이는 IRDS의 기능이 매우 다양하지만 시간과 비용이 많이 들기 때문에 IRDS의 핵심이 되는 기능 위주의 프로토타입을 설계하고자 한다. 또한 정보저장소의 분산 객체 환경을 지원하기 위한 방안으로 CORBA 미들웨어를 이용하고자 한다. CORBA 미들웨어는 네트워크 투명성을 지원하는 미들웨어로 분산 객체 환경을 지원할 수 있는 도구이다. 본 논문에서 제시하는 시스템의 설계 및 구현 방법은 서비스 인터페이스와 사용자 인터페이스에서 사용될 정보저장소 서비스 함수를 CORBA에서 제공하는 IDL로 정의하여, 이를 IDL 컴파일러에 의해 생성된 C++ 코드로 최종 구현하였다. 따라서 본 논문의 정보저장소는 CORBA가 제공하는 분산 객체 환경을 충분히 지원할 수 있다.

3.1 시스템 구조 설계

본 논문에서 분산 객체 환경을 위한 IRDS 기반의 정보저장소를 위해 제안한 구조는 (그림 1)과 같다.



(그림 1) 분산 객체 환경에서의 정보저장소 구조

전체 구조의 중심은 ORB이다. ORB는 분산 환경을 지원하는 일종의 미들웨어로서 워크플로우나 IRDS 사용자 인터페이스와 같은 IRDS 클라이언트와 IRDS 서버와의 통신을 담당한다. 즉, 클라이언트의 질의를 서버에 전달해주는 역할을 담당한다. IRDS 관리 서버는 네 개의 부분 즉, 질의 해석기, 질의 처리기, 일관성 검사기, 환경 설정기로 구성되며 데이터베이스 서비스 인터페이스를 통해 메타테이블에 접근한다.

- IRDS 사용자 인터페이스
  IRDS 사용자 인터페이스는 최종 사용자와 직접적으로 통신할 수 있는 사용자 단말 뷰로써 기본적으로 그래픽 사용자 인터페이스로 설계한다. 이는 IRDS 명령어에 익숙하지 않은 사용자에게 IRDS 명령어를 숨겨 주며 IRDS에 쉽게 접근할 수 있도록 IRDS 표준의 기능을 갖춘 인터페이스이다.
- 질의 해석기(Request Interpreter)
  ORB로 전달된 질의를 해석하여 정보저장소에 해당되는 함수로 변환한다. ORB로 전달된 질의는 일반적으로 workflow에서 사용되는 질의로 이를 직접적인 정보저장소에 해당하는 함수로 변환하기 위해서는 한 개 이상의 정보저장소 함수가 조합된 형태가 될 수 있으며 이를 하나의 트랜잭션으로 구성한다.
- 환경 설정기(Environment Seter)
  초기 시스템 환경을 설정하고 시스템을 이용하고 있는 사용자의 등록 및 관리를 담당한다. 환경 설정기는 처음 사용자 정보를 확인하고 사용자 등록일 경우 사용자에 맞는 환경을 설정한다. 환경 설정 과정은 사용자에 해당하는 메타테이블을 구성하고 각각에 해당하는 기본적인 메타데이터들을 로드시키는 과정이다.
- 일관성 검사기(Consistency Checker)
  전달된 명령어의 종류에 따라 메타데이터의 내용이 정당한지에 대한 일관성 및 무결성을 검사한다. 예를 들어, 개체 이름의 길이, 조작하려는 항목의 IRDS 내의 존재 여부, 시스템이 관할하는 사항의 수정 의도 등이 이 모듈에서 수행하는 사항이다. 연속적인 일관성 검사의 실행에 따라 오류 코드가 생성되며 하나의 오류라도 발생되면 질의의 수행을 중단하고 롤백한다.
- 질의 처리기(Request Processor)
  질의 해석기에 의해 변환된 정보저장소 명령을 처리하는 핵심 부분으로 각각의 질의를 다시 몇 개의 질의로 분할하여 일관성 검사를 거친 후 질의가 타당하다면 질의 처리를 시작한다.
- 데이터베이스 서비스 인터페이스(DBMS Service Interface)
  명령어들이 필요로 하는 모든 정보자원을 DBMS를 이용하여 필요한 서비스를 수행한다. 이는 일관성 및 무결성 검사에 대한 오류가 존재하지 않으면 질의를 수행하기 위해 필요한 SQL 명령어를 구성한다. SQL 명령어는 연속하는 몇 개의 SQL 명령어 집합이 될 수 있다.
- 메타테이블(Meta-table)
  메타데이터 정보를 내용별, 단계별로 체계화하여 저장하여 놓은 내부 정보 테이블로 단계별로 구분 가능하며 내부 테이블, 환경 테이블, 공통 테이블, 한정 테이블로 구성된다.

3.2 메타테이블 설계

IRDS의 메타테이블 설계 시 고려할 점은 두 단계의 데이터를 단계별 또는 단계간의 쌍으로 묶어 표현하는 것이다. 여기서 말하는 두 단계란 실세계 객체의 스키마인 메타데이터와 메타데이터의 스키마인 메타스키마를 말한다. ISO IRDS는 메타데이터를 IRD로 메타스키마를 IRD 정의로 명명하고 있으며, 두 단계 쌍들에 대해 최대한의 병렬성을 유지하면서 테이블의 내용에 따라 네 종류의 테이블로 구분하고 있다. 본 논문에서는 ISO IRDS 표준에 준하여 <표 1>과 같이 네 부류의 메타테이블을 설계하였다.

〈표 1〉 메타데이터 테이블

테이블 분류	테이블 설명	설계된 테이블
내부 테이블	• 한정 테이블의 각 객체에 대한 관리 및 버전 제어를 위해 설계된 테이블 • 각 IRD 단계의 한정 테이블과 IRD 정의 단계의 한정 테이블의 각 객체에 대해서 내부 테이블의 하나의 행으로 정의	IRD_OBJECT
환경 테이블	• IRDS 사용자, 구현 시 한계 값 등을 정의 • 각 IRD 정의마다 하나씩 존재	DICT_NAME
한정 테이블	• 자료 모델링 기능의 구조화 규칙을 표현하며 IRD 한정 테이블은 메타데이터를, IRD 정의 한정 테이블은 메타스키마를 관리 • IRD 한정 테이블과 IRD 정의 한정 테이블의 두 단계는 쌍을 이루며 타입/인스탄스 관계를 가짐 • 특정 IRD나 특정 IRD 정의에만 존재	IRD_DEF_TABLE IRD_DEF_COLUMN IRD_DEF_REL IRD_TABLE IRD_COLUMN IRD_REL
공통 테이블	• 모든 단계에서 공통으로 사용되는 테이블	MAT MET MRT

3.3 서비스 인터페이스 설계

IRDS 서비스 인터페이스는 외부 시스템이나 사용자가 IRDS 시스템에 접근하기 위한 내부 명령어라 할 수 있다. IRDS 서비스 인터페이스는 IRD 단계와 IRD 정의 단계에 기반한 서비스를 제공한다. IRD 단계의 서비스는 IRD 한정 테이블과 내부 테이블을, IRD 정의 단계의 서비스는 IRD 정의 한정 테이블과 내부 테이블을 조작한다. 또한 시스템 운영을 위해 IRDS 서비스를 처음 시작할 때는 IRDS 개방 서비스를, IRDS 서비스를 종료할 때는 IRDS 폐쇄 서비스를 사용한다. 사용자가 처음 등록할 경우에는 IRD 정의 생성 서비스를, 사용자 삭제 시는 IRD 정의 제거 서비스를 사용한다. IRD 단계에만 존재하는 서비스로는 IRD 생성 및 IRD 제거 서비스가 있다.

ISO IRDS 표준안을 근간으로 설계한 서비스 인터페이스는 시스템 운영에 관한 서비스, IRD 단계에 독립적인 서비스, IRD 정의 단계에 독립적인 서비스로 구분할 수 있다.

시스템 운영에 관한 서비스는 시스템의 활성화 및 종료 등 IRDS 시스템 운영에 관한 서비스로 IRDS 개방 서비스, IRDS 폐쇄 서비스, IRD 정의 생성 서비스, IRD 정의 제거 서비스가 있다. IRD 단계에 독립적인 서비스는 IRD 단계의 자료를 조작할 때 사용되는 서비스로 IRD 객체/관계 추가 서비스, IRD 객체/관계 삭제 서비스, IRD 객체/관계 수정 서비스, IRD 출력 서비스, IRD 생성 서비스, IRD 제거 서비스가 있다. IRD 정의 단계에 독립적인 서비스는 IRD 정의 단계의 자료를 조작할 때 사용되는 서비스로 IRD 정의 객체/관계 추가 서비스, IRD 정의 객체/관계 삭제 서비스, IRD 정의 객체/관계 수정 서비스, IRD 정의 출력 서비스가 있다.

3.4 사용자 인터페이스 설계

사용자 인터페이스는 IRDS를 탐색할 수 있는 그래픽 유틸리티이다. 즉, 시스템 명령어에 익숙하지 않은 사용자를 위해 시스템 명령어를 숨겨두고 시스템에 쉽게 접근할 수 있도록 표준적인 기능을 갖추도록 하며, IRDS 사용자는 사용자 인터페이스를 통해 IRDS 내의 객체나 타입의 설명을 볼 수 있고, 객체와 타입 간의 관계 및 객체와 객체 간의 관계 등을 볼 수 있다. 이를 위해 사용자 인터페이스를 IRD 단계 조작에 관한 명령어, IRD 정의 단계 조작에 관한 명령어, 유틸리티 명령어, 시스템 운영에 관한 명령어로 각각 구분하여 윈도우 GUI(Graphic User Interface) 방식을 토대로 설계하였다. 양질의 사용자 인터페이스를 제공하기 위해서 모든 명령어가 메뉴로 처리될 수 있도록 하향식 메뉴를 제공한다. 화면 구성은 기본적으로 특정 테이블을 선택할 수 있는 테이블 선택 영역, 각 명령어에 해당하는 매개변수를 입력할 수 있는 명령어 입력 영역, 실행 결과를 출력할 수 있는 결과 출력 영역, 오류 등의 결과 상태를 출력하는 메시지 영역, 시스템 사용법을 보여주는 도움말 영역으로 구분하여 설계하였다.

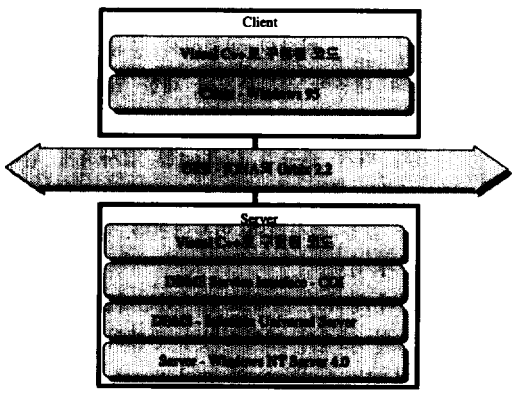
4. 분산객체 환경에서의 IRDS 기반 정보저장소 구현

분산객체 환경에서의 IRDS 기반의 정보저장소 구현은 기본적으로 ISO IRDS 표준안에 따라 구현하였다. 그러나 ISO IRDS 표준안의 구현 범위가 방대하고 모든 기능을 모두 구현하기에는 막대한 시간과 비용이 소모되므로 본 논문에서는 IRDS 표준안의 가장 핵심 기능이 되는 서비스 인터페이스와 사용자 인터페이스만 구현하였다. 서비스 인터페이스 기능 중 메타데이터의 버전 관리, 작업 관리, 보안 관리 등을 제외하였

다. 구현은 이미 설계한 시스템 구조에 맞게 서비스 인터페이스, 사용자 인터페이스를 구현하였으며 메타데이터 저장 공간인 메타테이블을 구현하였다.

4.1 시스템 구현 환경

정보저장소의 구현을 위해서는 크게 데이터베이스, 클라이언트 시스템, 서버 시스템, 상용 ORB 제품과 구현을 위한 툴이 필요하다. 구현에서 사용된 구현 환경은 (그림 2)와 같다.

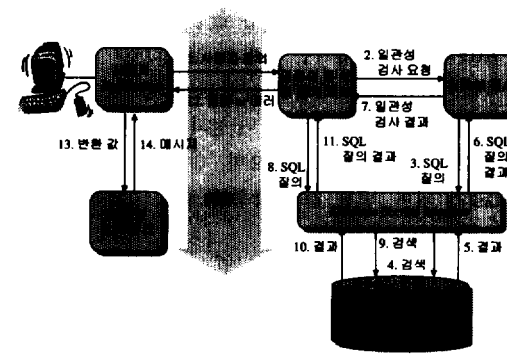


(그림 2) 시스템 구현 환경

클라이언트 시스템과 서버 시스템의 운영체제는 각각 Windows 95와 Windows NT 4.0을 사용하였다. 사용자 인터페이스의 구현 방식은 웹 기반 클라이언트, 윈도우 GUI 기반 클라이언트, 명령어 기반 클라이언트 등이 있는데 그 중 윈도우 GUI 기반 클라이언트로 구현하였다. ISO IRDS의 구현을 위한 데이터베이스 시스템은 객체 지향 데이터베이스가 적합하며, 이를 위해서는 IUS(Informix Universal Server)를 이용하였다[2,4]. IUS는 객체-관계형 데이터베이스로 객체와 클래스, 상속 등의 객체 지향 개념을 지원하며 본 논문의 구현을 위한 요구사항을 만족한다. 서버와 클라이언트 프로그램을 구현하기 위한 윈도우 프로그램으로 Visual C++을 사용하였다[13]. DBMS와 프로그램과의 연동을 위한 방법은 ODBC, ESQL, DBMS에 독립적인 연동 방식 등이 있는데, 여기서는 그 중 IUS에서 제공하는 COI(C++ Object Interface)를 이용하였다[3]. COI는 IUS에 접근하기 위해 C++에서 사용하는 일종의 라이브러리이다. CORBA를 지원하는 제품으로 현재 널리 상용화 되어 있는 것은 IONA의 Orbix와 Visigenics의 VisiBroker [16]등이 있으며 그 중 Orbix 2.2를 이용하였다[5].

4.2 시스템 구조 및 서비스 흐름 절차

분산객체 환경에서의 IRDS 기반 정보저장소의 구조는 IRDS 사용자 인터페이스, 질의 해석기, 질의 처리기, 일관성 검사기, 환경 설정기, 데이터베이스 서비스 인터페이스, 메타테이블로 구성되어 있다. 이러한 구조 하에서의 서비스 수행 절차는 (그림 3)과 같다.



(그림 3) 서비스 수행 절차

먼저 사용자 인터페이스를 통해 질의를 입력 받은 클라이언트의 사용자 질의는 ORB를 통해 서버로 보내진다. 서버는 질의해석기를 통해 질의를 해석하며, 질의 처리기를 통해 명령어를 처리한다. 처리 과정에 앞서 일관성 검사기를 통해 질의에 대한 일관성 및 무결성을 계속적으로 검사한다. 일관성 검사는 데이터베이스 서비스 인터페이스를 통해 메타테이블을 검색하면서 이루어진다. 명령어의 일관성과 무결성에 하나라도 결함이 있으면 오류 메시지를 반환하고 그렇지 않으면 주어진 명령어를 역시 데이터베이스 서비스 인터페이스를 통해 메타테이블을 조작하면서 수행한다. 결국 서버는 질의에 대한 오류 번호나 성공 메시지를 사용자에게 반환하게 되는데 사용자 응용 프로그램은 오류 번호이면 번호에 해당하는 적당한 메시지를 출력하고 성공적인 질의이면 성공 메시지를 출력한다.

4.3 서비스 인터페이스 구현

서비스 인터페이스 구현을 위해서 먼저 CORBA IDL로 정의된 인터페이스를 C++로 바인딩된 각각의 함수에 각 기능을 위한 코드와 IUS에서 제공하는 데

이터베이스 서비스 인터페이스를 사용해 구현하였다. 서비스 인터페이스는 ISO IRDS 표준안에 구현함에 따라 사용자 정의 타입과 몇 개의 함수로 구성된다. 구현에서 정의한 사용자 정의 타입은 서비스 인터페이스와 자료 값을 주고 받는데 사용되는 열 목록 매개변수의 구조를 정의하는 구조체 변수로 ColList를 정의하였다. ColList의 구조는 ColName, ColType, ColValue로 구성되며 ColName은 매개변수로 전달될 각 열의 이름, ColType은 각 열의 자료타입, ColValue는 ColName에 해당하는 값을 나타낸다. 서비스 함수는 이미 설계한 대로 크게 시스템 운영에 관한 서비스 함수, IRD 단계에 독립적인 서비스 함수, IRD 정의 단계에 독립적인 서비스 함수로 구분된다. 함수는 함수 이름, 매개변수, 반환 값으로 구성되는데 함수 이름은 ISO IRDS 표준을 참조하였으며, 대부분의 매개변수는 ColList가 되며, 반환 값은 함수의 수행 결과를 나타내는 오류 코드이다.

- 질의 해석기

  질의 해석기는 ORB로 전달된 사용자 인터페이스의 질의를 해석하여 정보저장소에 해당되는 함수로 변환한다. 그러나 대부분의 클라이언트 질의는 하나의 함수로 대응되며 함수 내의 여러 개의 부함수로 구성된다. 예를 들어, 클라이언트에서 새로운 사용자를 등록하라는 질의인 IrdsCreateIRDDefinition을 호출하면 서버의 IrdsCreateIRDDefinition은 공통 테이블인 MAT, MET, MRT를 생성하고, IRD 정의 단계의 테이블인 IRD_DEF_TABLE, IRD_DEF_COLUMN, IRD_DEF_REL을 생성하고, IRD 단계의 테이블인 IRD_TABLE, IRD_COLUMN, IRD_REL을 생성하고, 내부 테이블인 IRD_OBJECT를 생성하는 함수를 차례로 호출하며, 일관성 검사기에 질의를 전달한다.

- 환경 설정기

  환경 설정기는 처음 사용자가 로그인 해서 사용자 등록이 되어 있지 않을 경우, 사용자 등록을 원할 때 실행된다. 사용자 처음 로그인 하는 함수는 IrdsOpen이고, 이 경우 사용자가 등록되었으면 사용자의 세션 ID를 반환하고, 그렇지 않으면 오류 값을 반환한다. 만약 오류 값이 반환되면 사용자 등록을 요구하고, 사용자 등록을 원한다면 IrdsCreateIRDDefinition을 호출한다. 이 함수는 등록할 사용자에게 새로운 세션 ID를 부여하고 사용자 정보와 함께 DICT_NAME 테이블에 등록한다. 이 후 IRD 정의 단계의 테이블인 IRD_DEF_TABLE, IRD_DEF_COLUMN, IRD_DEF_REL과 IRD 단계의 테이블인 IRD_TABLE, IRD_COLUMN, IRD_REL과 내부 테이블인 IRD_OBJECT를 생성하고 기본적인 데이터를 로드시킨다. 사용자 삭제할 경우에는 IrdsRemoveIRDDefinition를 호출하고 매개변수인 사용자 세션에 해당하는 사용자를 DICT_NAME서 삭제하며, 관련된 테이블을 모두 삭제한다.

- 일관성 검사기

  일관성 검사기는 전달된 명령어의 종류에 따라 메타데이터의 내용이 정당한지에 대한 일관성 및 무결성을 검사한다. 즉, 사용자 인터페이스로 받아 들인 질의의 매개 변수인 ColList를 검사하게 된다. 검사는 적당한 'SELECT" 문을 생성하고 "SELECT" 문의 결과 값을 검사하여 이루어진다. 검사 항목의 예는 다음과 같다.

  - ColList의 항목이 제대로 들어있는지를 검사한다. IRD 정의 단계의 메타스키마 관계의 추가일 경우 ColList의 ColName에 me1_name, me2_name, me_rel_name과 이에 해당하는 값(ColValue)은 반드시 있어야 한다. me1_name과 me2_name은 관계를 연결하기 위한 두 개의 메타스키마 이름이고, me_rel_name은 관계의 이름이다.
  - ColList의 ColValue의 항목이 정당한지를 검사한다. IRD 정의 단계의 메타스키마 관계의 추가일 경우 me1_name과 me2_name은 IRD_DEF_TABLE에 존재하는 값 중의 하나이어야 한다. 이러한 경우의 "SELECT" 문은 "select me_name from IRD_DEF_TABLE where me_name = ?"이고 ?는 ColList의 ColValue에 해당하는 값이다. 이러한 "SELECT" 문의 결과가 존재하면 일관성 검사가 정당한 것이고 그렇지 않으면 오류이며, 이 때는 오류 번호를 반환하고 실행을 즉각적으로 중단하고 롤백한다.

- 질의 처리기

  질의 해석기에 의해 적당한 함수를 구성하고 일관성 검사기에 의해 질의가 정당하다면 질의 처리를 시작한다. 질의 처리는 해당 명령어에 맞는 SQL 문을 생성하고 이를 적절한 변수에 저장한다. 저장된 변수는 COI에서 제공하는 SQL 질의 처리 함수의

매개변수로 사용되어 이를 실행한다. 실행이 성공적으로 끝나면 SQL 질의 처리 함수는 질의의 결과를 반환하며, 질의 결과에 대해 오류이면 오류 번호를, 성공이면 결과 값을 반환한다. 예를 들어, IRD 정의 단계의 메타스키마 관계의 추가일 경우 두 개의 테이블에 접근하기 위한 질의 즉, IRD_DEF_REL과 IRD_OBJECT에 객체를 삽입하는 SQL 질의를 생성하면 되고, 객체는 ColList의 ColName과 ColValue, ColType을 참고하여 객체의 속성을 구성한다. 객체 추가 시 객체의 키는 IRD_DEF_REL을 검색하여 가장 큰 키 값에 1을 더한 값으로 한다.

서비스 인터페이스를 구현하기 위한 핵심적인 IRDS 기반의 함수의 일부 예는 <표 2>와 같다. 함수는 시스템 운영에 관한 함수, IRD 정의 단계에 독립적인 함수, IRD 단계에 독립적인 함수로 구분하여 구현했으며, 함수명과 매개변수, 반환 값을 명시해 놓았다. 매개변수 중 login_info는 사용자 로그인 정보를, ColList는 자료를 주고 받는데 사용되는 열 목록을 정의하는 사용자 정의 타입이다. 반환 값은 오류를 나타내는 값으로 모든 함수의 반환 값이기도 한다. <표 2>에서 제시하는 함수의 프로토타입은 CORBA IDL로 정의한 것이다.

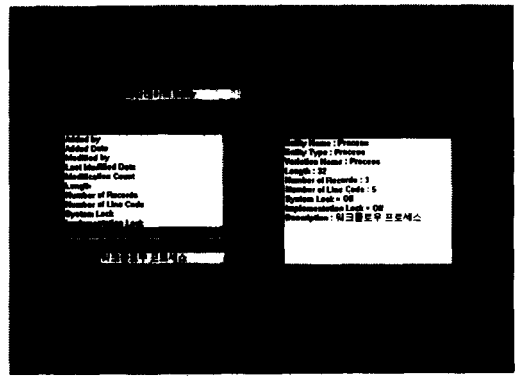
〈표 2〉 IRDS 함수의 예

함수 분류	구현된 함수
시스템 운영에 관한 서비스 함수	long IrdsOpen(in login_info li, out short CurrSessId)
	long IrdsClose(out short CurrSessId)
	long CreateIRDDefinition(in login_info li, out short CurrSessId)
	long IrdsRemoveIRDDefinition(in short CurrSessId)

4.4 사용자 인터페이스 구현

사용자 인터페이스의 구현은 설계 목표에 따라 Visual C++를 이용하여 윈도우 기반 GUI 방식 인터페이스로 구현하였다. 이 중 사용자 인터페이스 전체 메뉴를 포함한 메타스키마 추가를 위한 입력 창의 예는 (그림 4)와 같다.

사용자 인터페이스 메뉴는 크게 시스템 운영, 메타스키마 조작, 메타데이터 조작, 메타데이터 뷰, 유틸리티, 도움말로 구성된다. 시스템 운영 메뉴를 선택하면 시스템 개방, 시스템 종료가 나타난다. 시스템 개방을 선택하면 사용자 로그인 정보를 입력하기 위한 창이 나타난다. 시스템 종료를 선택하면 확인 메시지 후 시스템을 종료한다. 메타스키마 조작 메뉴를 선택하면 메타스키마 추가, 삭제, 수정 메뉴가 나타나며, 메타데이터 조작 메뉴를 선택하면 메타데이터 추가, 삭제, 수정, 메뉴가 나타난다. 메타데이터 뷰 메뉴는 메타데이터/메타스키마의 내용을 관련된 관계 정보와 함께 보여주고, 유틸리티는 IRD 생성과 삭제를 위한 메뉴이다. 여러 사용자 인터페이스 중 메타스키마 추가를 위한 인터페이스를 예를 들어, 메타스키마 추가 인터페이스 화면은 테이블 선택을 위한 콤보 박스, 선택된 테이블의 추가할 속성을 입력 받기 위한 리스트 박스, 추가한 결과를 보여주는 리스트 박스, 추가할 속성 값을 입력 받는 콤보 박스로 구성된다. 다른 사용자 인터페이스도 각각의 특성에 맞게 구성하였다.



(그림 4) 사용자 인터페이스 예

5. 분산객체 환경에서의 IRDS 기반 정보저장소의 워크플로우에의 활용

이미 완성된 분산 객체 환경에서의 IRDS 기반 정보저장소를 검증하기 위하여 워크플로우 관리 시스템에 적용하였다. 이를 위해서는 워크플로우에 사용되는 데이터의 수집 및 분석이 필요하다. 워크플로우 데이터는 메타데이터의 스키마인 메타스키마와 응용데이터의 스키마인 메타데이터로 나누어 분석되어야 한다. 메타스키마는 메타데이터인 IRD 단계에 있는 정보를 나타내기 위한 수단을 제공하며 IRD 단계의 정보의 타입을 결정한다. 메타데이터는 응용데이터베이스에 대한 스키마이면서 메타스키마가 가지는 타입의 인스탄스이며 응용 데이터베이스 정보를 나타내기 위한 타입을

제공한다. 이와 같이 분석된 결과를 가지고 IRDS에서 제공하는 모델링 방법에 따라 통합적인 메타모델링이 이루어져야 하는데, 모델링 또한 메타데이터인 IRD 단계의 메타모델링과 메타스키마인 IRD 정의 단계의 메타모델링이 필요하며 두 단계의 데이터는 타입과 인스탄스의 관계를 가지고 있어야 한다. 메타모델링에 정의된 데이터들이 메타테이블에서 관리되는데 메타테이블의 내용은 메타테이블의 생성 시 기존에 이미 정의된 데이터가 있으며, IRDS 사용자가 특정 응용을 위해 정의한 데이터가 있다.

5.1 IRD 정의 단계의 메타모델링

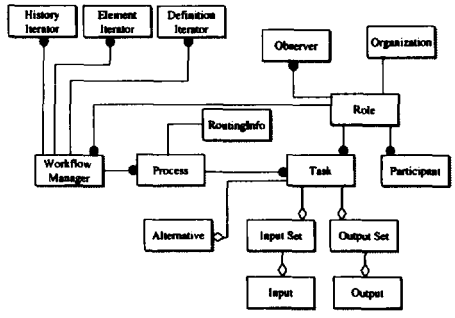
이 단계는 IRD 정의를 위한 단계로 하나 이상의 IRD 정의를 기술하고, 하나 이상의 IRD 정의는 하나 이상의 IRD를 포함한다. 또한 IRD 단계에 있는 정보를 나타내는 수단을 제공하기 위해 IRD에 대한 타입을 정의하여 관리한다. 이러한 타입은 IRD 정의 단계 테이블의 인스탄스로 관리된다[14]. 예를 들어, HistoryIterator, ElementIterator, DefinitionIterator라는 IRD 단계의 데이터의 타입은 Iterator가 되고, WorkflowManager, Observer, Participant라는 데이터의 타입은 User가 될 수 있다. 이처럼 IRD 단계에 나타나는 인스탄스의 타입을 하나의 인스탄스로 고려하여 관리하는 것이 IRD 정의 단계의 테이블이다. IRD 정의 단계에 저장될 데이터들은 메타모델링에 따라 각 테이블에 저장된다.

IRD 정의 단계에서 워크플로우를 위해 관리되는 메타스키마들은 프로세스 정의를 위한 ProcessDefinition, 워크플로우 관련 데이터인 WfRelevant, 워크플로우 제어를 위한 데이터인 WfControl, 조직체 정보를 위한 Organization, 사용자와 태스크간의 역할을 위한 Role, 작업 단위를 나타내는 WorkList, 반복적인 데이터의 사용을 위한 Iterator, 워크플로우 사용자를 위한 User가 존재한다.

5.2 IRD 단계의 메타모델링

IRD 단계에서는 IRD들이 저장되며 IRD 내에 저장되는 데이터의 타입은 IRD 정의 단계에서 반드시 기술되어 있어야 한다. 즉, IRD 단계의 데이터는 일반 데이터베이스에 저장되는 인스탄스에 대한 스키마이며 IRD 정의 단계의 인스탄스를 타입으로 가지는 데이터들이다[14]. IRD 단계에서 워크플로우를 위해 관리되는 메타데이터들을 IRD 정의 단계의 인스탄스를 대상으로 한 모델링은 (그림 5)와 같다. (그림 5)의 모델의 중심은 Process이다. Process는 워크플로우의 작업 단위로 여러 개의 작은 작업 단위인 복수 개의 Task로 구성된다. Task는 Task의 참여자와 Task 간의 의존관계를 나타내는 Alternative, Task의 입출력 집합을 나타내는 InputSet, OutputSet으로 구성된다. 워크플로우를 위한 사용자는 전체 워크플로우를 관장하는 WorkflowManager, 특정 Task에 참여하는 Participant, 워크플로우의 흐름을 감시하는 Observer가 있으며 이들 모두 고유의 Role을 가진다. Iterator는 워크플로우 진행 중 반복적인 데이터로서, HistroyIterator는 워크플로우의 반복적인 이력 정보를, ElementIterator는 워크플로우 실행을 위한 구성 요소를 반복적으로 접근하기 위한 정보를, DefinitionIterator는 워크플로우 프로세스 정의를 위한 반복적인 정의 정보를 나타낸다. 이외 조직체 정보를 나타내는 Organization과 사용자의 워크플로우에 대한 임무를 나타내는 Role이 있다.

IRD 단계의 데이터들을 관리하는 테이블은 IRD_TABLE, IRD_COLUMN, IRD_REL이 있다. (그림 5)의 워크플로우의 데이터는 IRD 단계의 세 테이블에 적당한 형태의 인스탄스로 관리된다. 예를 들어, 전체 객체 중 WorkflowManager, Process, Task 세 개의 객체만을 대상으로 실제 테이블에서 어떤 식으로 관리되는지 살펴보면, 일단 IRD_TABLE에는 세 개의 객체 즉, WorkflowManager, Process, Task가 추가되며, IRD_REL에는 세 객체의 관계 즉, WorkflowManager-Contains-Process, Process-Contains-Task가 추가된다. 또한 IRD_COLUMN에는 세 객체의 속성들 예를 들어, Process일 경우 ProcessNo, ProcessPriority, ProcessVersionNo, ProcessName 등이 추가된다. 또한 추가된 각 행에 대해 객체 관리를 위해 IRD_OBJECT에 한 행이 추가된다.



(그림 5) 워크플로우를 위한 IRD 단계의 메타 모델

IRD 정의 단계와 IRD 단계는 타입/인스탄스 관계가 있다. <표 3>은 IRD 단계와 IRD 정의 단계의 데이터에 대한 타입/인스탄스의 관계의 예이다. IRD_DEF_TABLE과 IRD_TABLE의 인스탄스에 대해서는 모두 기술하며, IRD_DEF_REL과 IRD_REL에 대해서는 그 양이 많으므로 User와 ProcessDefinition, User와 Role에 한하여 부분적으로 기술하였다.

〈표 3〉 IRD 단계와 IRD 정의 단계의 타입/인스탄스 관계

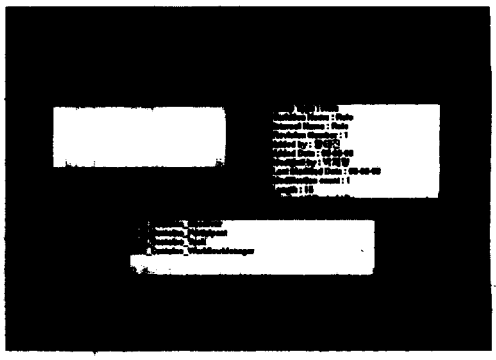
	IRD 정의 단계 인스탄스	IRD 단계 인스탄스
IRD_DEF_TABLE /IRD_TABLE	ProcessDefinition	Process Task
	WfRelevant	Alternative RoutingInfo
	WfControl	현재 없음
	User	WorkflowManager Observer Participant
	Iterator	HistoryIterator ElementIterator DefinitionIterator
	Worklist	InputSet OutputSet Input Output
	Organization	Organization
	Role	Role
IRD_DEF_REL /IRDREL	User-Related-to-ProcessDefinition	WorkflowManager-Contains-Process
	User-Related-to-Role	Role-Contains-Observer Role-Contains-Participant

5.3 워크플로우를 위한 정보저장소 검증

정보저장소가 워크플로우 응용을 위해서 어떠한 식으로 메타데이터를 관리하는지를 예제를 통해 서비스 인터페이스와 관련된 테이블을 가지고 설명하며, 그 결과가 사용자 인터페이스로 어떻게 보여지는지 기술한다. 예제는 IRD 단계에서 처리되는 예와 IRD 정의 단계에서 처리되는 예를 보인다.

[예제] "Role 테이블을 삭제할 경우 Role 테이블과 관련된 테이블을 찾아라"

어떠한 경우에는 IRD 단계의 일부의 테이블이 더 이상 사용되지 않거나 아니면 변경을 할 경우가 생긴다. 즉, (그림 5)의 Role테이블을 아예 삭제하거나 테이블 이름을 변경할 수 있을 것이며, 이럴 경우 Role 테이블과 관련된 모든 테이블과 관계들에 변경이 초래된다. 이런 경우, IRDS 관리자는 Role 테이블에 관한 정보를 얻고 Role 테이블과 연관된 모든 관계들의 정보를 파악하여야 Role 테이블 변경에 따른 파급효과를 미리 감지할 수 있다. 이럴 경우에는 IRDS 사용자는 사용자 인터페이스를 통해 테이블 입력 창에서 'IRD_TABLE'을 입력하고, 'Role'과 같은 출력할 객체 이름을 입력하고, 출력하여 보고자 하는 속성을 선택하고 확인 버튼을 누른다. 시스템은 Output이라는 서비스 인터페이스 함수를 통해 입력한 매개변수를 넘겨주고, 서버의 질의 해석 및 일관성 검사, 질의 처리를 통해 관련된 모든 정보를 넘겨 받는다. (그림 6)은 [예제]의 결과 화면이다. Output 서비스를 수행하여 결과를 보여주는 화면은 크게 세 부분으로 나누어진다. 하나는 화면 왼쪽 위 부분의 엔티티를 표시하는 부분이고 또 하나는 화면 오른 쪽 위 부분의 엔티티에 관한 일반적인 정보를 표시하는 부분이고, 또 하나는 화면 아래 쪽의 엔티티와 관련된 모든 관계를 보여주는 부분이다. Role의 엔티티에 관한 정보는 엔티티 이름, 변형 이름, 내부 이름인 Role과 Role 엔티티의 타입이며 IRD 정의 단계의 한 인스탄스인 Role, Role 엔티티를 처음 생성한 시간(09-08-98)과 생성한 사람(염태진), 마지막으로 수정한 시간(09-08-98)과 수정한 사람(박재형), Role 엔티티의 내부 길이(16), 레코드 수(4) 등이 나온다. Role 엔티티와 관련된 관계는 Role-Contains-Observer, Role-Contains-Participant, Role-Contains-Task, Role-Contains-WorkflowManager이며 결국 Role 테이블을 변경했을 경우, 관련된 네 개의 관계와 네 개의 엔티티(Observer, Participant, Task, WorkflowManager)에 변경을 가해야 한다는 점을 알 수 있다.



(그림 6) [예제]의 결과

6. 성능분석

이번 장에서는 논문에서 제안하고 구현한 시스템과 기

존의 정보저장소인 UNISYS의 UREP, IBM의 Repository Manager/MVS와 비교 분석한다.

UREP는 OMG에서 제안한 분산 환경에서의 정보저장소를 기반으로 UNISYS사에서 상용 제품으로 만든 정보저장소다. UREP은 3단계의 저장 구조에 데이터를 저장하고 이를 제한된 분산 환경에서 운용 가능하도록 되어 있다. 3단계란 구조는 정보저장소의 가장 기본적인 개념인 모델(혹은 스키마), 클래스, 관계, 사건 등을 기술하는 정보저장소 객체 모델, 정보저장소 공통 기능을 나타내는 정보저장소 서비스 모델, 정보저장소 내용을 기술하는 기술/도구/비즈니스 모델 의 구조를 말한다. UREP은 분산 환경에서 동적으로 작동하며, 여러 가지 도구와 통합 가능하며, 다양한 사용자 인터페이스, 응용 프로그램 등을 지원한다. 그러나 UREP는 현재 진정한 분산 환경이라 볼 수 없고 제한적으로 CORBA가 아닌 RPC로 작동한다.

Repository Manager/MVS는 IBM이 개발한 응용 시스템개발 환경인 AD/Cycle의 모든 생명주기 단계를 지원하는 소프트웨어 도구들과 서비스를 제공하는 핵심 부분이다. 이러한 서비스들의 확장 집합은 정보저장소와 인터페이스할 수 있도록 제공해 준다. 정보저장소 내에 저장된 정보는 임의의 도구에 의해 접근되어지고 Repository Manager/MVS에 의해 표현되는 형식에 의해 조작된다. 결국 이러한 정보는 서로 다른 도구들간에 정보공유가 가능해 진다. SAA(System Application Architecture)의 한 요소인 Repository Manager/MVS는 SAA 환경 전체에 걸쳐 포괄적이고 통합된 응용시스템 개발의 틀을 제공하는 기반이 된다. 이러한 SAA 환경은 생명주기 전체를 지원, 공통된 사용자 인터페이스, 공통된 정보저장소, 개방형 구조를 제공한다.

본 논문에서 구현한 시스템은 ISO IRDS에 기반하여 구현된 시스템으로 분산 환경에서 자유롭게 운영될 수 있도록 CORBA 기반으로 되어 있다. 따라서 분산 환경에서 서로 다른 이질적인 시스템들이 정보저장소를 공유하며, 시스템들이 서로 연계되어 사용될 수 있다. 또한 본 시스템은 사용자 편의를 위해 윈도우 기반의 GUI를 지향한다.

정확한 비교 분석을 위해 각 시스템의 데이터 관리 방법, 모델링 방법, 정보저장소 각 기능, 분산 환경 지원 가능성, 이질 환경 지원 가능성, 타 시스템과의 연계성, 사용자 인터페이스 등을 비교하였다. <표 4>는 본 논문에서 구현한 분산 환경에서의 정보저장소와 UNISYS의 UREP, IBM의 Repository Manager/MVS 세 시스템의 비교한 결과이다.

〈표 4〉 정보저장소 시스템 비교

비교대상	분산객체 환경에서의 IRDS 기반 정보저장소	UREP	Repository Manager /MVS
데이터 관리	3 layer fundamental layer, meta-schema layer, meta-data layer	3 layer conceptual model, service model, business model	3 view conceptual view logical view storage view
모델링 방법	객체지향 모델링	객체지향 모델링	객체지향 모델링
메타데이터 관리 및 모델링 기능	가능	가능	가능
생명주기 관리 기능	불가능	가능	가능
버전 관리 기능	불가능	가능	가능
보안 기능	불가능	가능	가능
분산 환경	가능	제한적으로 가능	불가능
사용자 인터페이스	GUI	GUI	GUI
다른 시스템과의 연계 가능성	가능	불가능	불가능
이질 환경	가능	불가능	불가능

7. 결 론

조직체 내의 각 응용 시스템들이 요구하고 사용하는 모든 정보 자원을 효율적으로 저장, 관리, 통합하여 공유 가능하게 지원하는 시스템에 대한 요구는 정보 처리의 속도가 빨라지고 그 양이 방대해짐에 따라 날로 증가하고 있다. 이러한 요구에 대한 해결책으로 정보저장소라는 개념이 등장하고, 정보저장소에 대한 표준안 연구 및 상품이 나오고 있다[14,17]. 이러한 표준 중의 하나가 본 논문에서 구현한 IRDS이다. 그러나 IRDS는 분산 객체 환경에서는 여러 시스템과 연동하여 적용하기가 어려웠다.

이러한 요구사항에 따라 본 논문에서는 ISO의 IRDS를 기반으로 분산 객체 환경에서 운영이 가능한 정보저장소 프로토타입을 설계 및 구현하였다. 원활한 구현을 위해 정보저장소의 핵심 DBMS는 Informix Universal Server 객체 관계용 데이터베이스를 사용하였고, 분산 객체 환경을 지원하기 위해 IONA의 Orbix 2.2를 사용하였다. 또한 사용자에게 친숙한 GUI 인터페이스를 제공하기 위해 Visual C++을 이용한 사용자

인터페이스도 구현하였다.

본 논문에서 구현한 정보저장소는 기능면에서 ISO IRDS 표준안에 있는 모든 기능은 구현하지 못했지만 핵심 기능을 가진 정보저장소 프로토타입을 구현하였으므로, 프로토타입의 활용도는 크다고 할 수 있다. 실제 워크플로우 관리 시스템에 정보저장소를 활용한 결과와 그 결과를 검증해 보았으며, 워크플로우 이외에도 CORBA를 이용한 분산 객체 환경에서 여러 시스템들과 연계하여 직접 사용할 수 있도록 서비스 인터페이스를 몇 개의 함수로 모듈화하였다.

그러나, 원래가 정보저장소의 프로토타입을 구현할 목적이었으므로, 버전 관리 기능, 보안 기능 등은 제외한 정보저장소의 핵심 기능만을 구현하였다. 향후 연구 방향으로 데이터의 변경에 대해 능동적으로 대처할 수 있는 버전 관리 기능, 인가된 사용자만이 정보자원을 접근할 수 있고 데이터에 대한 보호를 위한 보안 기능 등이 요구되며, 또한 워크플로우 응용을 위해서는 다수의 워크플로우 엔진을 사용하면서, 그들간의 협력 체제가 이루어 질 수 있도록 정보 자원을 관리할 수 있는 기능이 요구된다. 이를 위해서는 자원에 대한 독립적인 위치를 보장하면서 네트워크를 주소를 사용자 이름으로 변환해주고 중복된 목록을 제공해 줄 수 있는 기능 등의 목록 관리 기능이 필요하다.

참 고 문 헌

[1] David Hollingsworth, Workflow Management Coalition The Workflow Reference Model, pp.2-15, Workflow Management Coalition, 1994.

[2] Informix, Informix Error Message, Informix Inc., 1997.

[3] Informix, INFORMIX FastStart for University, Informix Inc., 1997.

[4] Informix, Informix Universal ServerInformix Guide to SQL, Informix Inc., 1997.

[5] IONA Inc., Orbix 2 Programming Guide, IONA Inc., 1995.

[6] James Rumbaugh, Object-Oriented Modeling and Design, Prentice Hall International Inc., 1991.

[7] Mark Jones, "Brave New World : A Vision of IRDS," pp.43-49, Database Programming and Design, Nov, 1991.

[8] Mark R. Jones. "Unveiling Repository Technology", pp.28-35, Database Programming and Design, April, 1992.

[9] Nortel, Workflow Management Facility Specification, pp.48-55, OMG, 1997.

[10] Randy Otte, Understanding CORBA, Prentice Hall International Inc., 1996.

[11] Shunsuke Akifuji, Workflow Management Facility, OMG, 1997.

[12] Thomas J. Mowbray, The Essential CORBA : Systems Integration Using Distributed Objects, John Wiley & Sons Inc., 1995.

[13] 곽준기, 백정렬, Inside Secret Visual C++ 5.0, 삼각형, 1997.

[14] 김기봉, "IRDS 기반 정보저장소와 CASE 도구 통합에의 활용", 박사 학위 논문, 충남대학교, 1998.

[15] 김기봉, 박중기, 조유희, 진성일, "정보자원 관리를 위한 IRDS 서비스 사용자 인터페이스의 설계 및 구현", 정보과학회논문지, 제22권 제11호, pp.1499-1509, 1995. 11.

[16] 박재현, Core CORBA 코아 코바, 영한출판사, 1998

[17] 진성일 외, 통합자료관리체계 시험구현연구 최종보고서, 국방과학연구소, pp.1-9, 1998.

[18] 한국전산원, 정보자원사전시스템(IRDS) 서비스 표준 연구, 한국전산원.

염 태 진

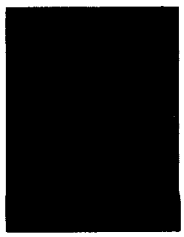
e-mail : tjyeom@sorec.chungnam.ac.kr

1997년 충남대학교 컴퓨터과학과 졸업(학사)

1999년 충남대학교 대학원 컴퓨터과학과 졸업(이학석사)

1999년~현재 충남대학교 소프트웨어연구센터 전임연구원

관심분야 : 통합데이터베이스, CALS/EC, 정보저장소, 객체지향, Workflow 등



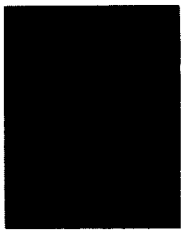
박 재 형

e-mail : jhpark@cs.chungnam.ac.kr

1998년 충남대학교 컴퓨터과학과 졸업(학사)

1998년~현재 충남대학교 대학원 컴퓨터과학과 석사과정

관심분야 : 데이터베이스, 통합데이터베이스, 멀티미디어시스템, 분산객체시스템, 공동작업 시스템 등



리 자

e-mail : lizi@sorec.chungnam.ac.kr

1992년 중국 길림대학교 컴퓨터과학과(학사)

1999년 현재 충남대학교 컴퓨터과학과 대학원 석사과정

관심분야 : 데이터베이스, 통합데이터베이스, 분산객체시스템, 버전제어시스템 등

김 기 봉

e-mail : kbkim@tiger.tjhealth.ac.kr

1991년 충남대학교 컴퓨터과학과 졸업(학사)

1993년 충남대학교 대학원 컴퓨터과학과 졸업(이학석사)

1998년 8월 충남대학교 대학원 컴퓨터과학과(이학박사)

1994년 9월~1997년 2월 혜전대학 전산정보처리과 교수

1997년 3월~현재 대전보건대학 전산정보처리과 교수

관심분야 : 통합데이터베이스, 정보저장소, CALS/EC 등

진 성 일

e-mail : sijin@cs.chunanam.ac.kr

1978년 서울대학교 계산통계학과 (학사)

1980년 한국과학기술원 전산학과 (이학석사)

1994년 한국과학기술원 전산학과 (이학박사)

1983년~현재 충남대학교 컴퓨터과학과 교수

관심분야 : 데이터베이스, 멀티미디어시스템, 소프트웨어공학, 시뮬레이션 모델링 등