객체그룹 기반의 분산환경에서 교육용 VDB 시스템의 설계 및 구현

유 경 택[†] · 이 현 철[††] · 주 수 종[†††]

요 약

분산환경에서 효율적인 멀티미디어 서비스를 제공하기 위해 TINA 콘소시엄과 OMG CORBA에서는 객체지향기술을 적용한 분산 객체 플랫폼의 제안과 다양한 서비스의 요구사항들을 정립하고 있다. 그러나 멀티미디어 서비스 어플리케이션의 규모가 점점 커지고 분산화 됨에 따라 객체들간의 서비스 및 관리 인터페이스가 매우 복잡해지고 있다. 이러한 문제들을 해결하기 위해서 새로운 객체그룹 모델의 제안과 객체그룹 하에서 도입될 수 있는 객체 관리 및 서비스를 위한 요구사항들이 필요하다. 그 동안 우리는 개개의 객체들을 관련 서비스 별로 그룹핑 하고 분산 객체 및 객체그룹들간의 접속을 위한 트레이딩 기능을 제공할 수 있는 분산 객체그룹 플랫폼을 개발하여 왔다[12-14]. 논문에서는 이러한 분산 객체그룹 플랫폼 상에 원격 교육 서비스용 VDB(Virtual Drawing Board)를 설계 및 구현하였다. 결과로서 교육 서비스용 분산 객체 및 객체그룹들로 구성된 VDB 시스템의 기본 구조와 서비스 인터페이스를 설계하고, 이를 통해 구현한 시스템의 수행과정을 보였다. 본 시스템의 분산 서비스를 지원을 위해 객체 미들웨어로서 CORBA 제품인 IONA사의 Orbix 2.2, 분산 객체들간의 접속을 위한 OrbixTrader 1.0과 분산 객체그룹들의 접속 및 그룹내의 객체들의 접근 권한을 조사할 수 있도록 자체 개발한 OGTG를 사용하였다.

Design and Implementation of An Educational VDB System in Distributed Environments Based on Object Groups

Kyeong-Taek Rhyu[†] · Hyun-Chul Lee[††] · Su-Chong Joo[†††]

ABSTRACT

For efficiently providing multimedia services, distributed computing environments are specified the requirements of various services and distributed object platforms applied an object-oriented technology by TINA Consortium and OMG CORBA. Because multimedia service applications are becoming large and distributing, their servicing and managing interfaces among objects are being complicated. In order to solve these defects, it is necessary to suggest a new object grouping model and specify object service/management requirements can be introduced under the object groups. we have been developed the distributed object group platform that can group all individual objects by the relating services and can supply trading functions for interconnecting between distributed objects or object groups.

In this paper, we designed and implemented the Virtual Drawing Board for remote equational services on the distributed object group platform we mentioned above. As results, we designed a basic structure and service interfaces, and showed execution procedures of VDB system consisted of distributed objects and objects groups for educational services. For supporting distributed services of VDB system, we used three kinds of tools as follows ; IONA Orbix 2.2 of CORBA compliance as an object middleware, OrbixTrader 1.0 for interconnection of distributed objects, and the OGTG we developed for interconnection of distributed object groups and checking access rights of objects included in an arbitrary object group.

† 종신회원 : 극동정보대학 전산정보처리과 교수
†† 준 회 원 : 원광대학교 대학원 컴퓨터공학과
††† 정 회 원 : 원광대학교 컴퓨터공학과 교수
논문접수 : 1999년 1월 25일, 심사완료 : 1999년 10월 5일

1. 서 론

이 기종 분산환경에서 날로 복잡해지는 분산 어플리케이션 서비스 플랫폼으로써, 새로운 개방형 통신망의 구조 정립을 위해서 객체지향 모델링 기법을 적용한 개방형 정보 통신망 구조의 연구[1, 7, 8, 15]가 활발하게 진행 중에 있다. 분산 객체 시스템은 전형적으로 분산 컴퓨팅 노드들 상에 펼쳐진 객체들의 인스턴스들과 바인딩되는 객체 인스턴스들의 분산된 집합으로 구성되므로 그 관리 및 서비스가 매우 복잡하다. 따라서 개방형 통신망인 TINA-C(Telecommunication Information Network Architecture-Consortium)의 TINA 권고안에서는 개별적인 객체뿐만 아니라, 객체들의 집합을 관리하는 구조화 메카니즘으로써 객체그룹(Object Group)을 정의하고 있다[1, 2, 6]. 객체그룹화의 목적은 다양하고 복잡한 분산 멀티미디어 서비스를 단순하게 관리하고 서비스를 지원하는데 있다. 이러한 목적을 위해서 우리는 객체그룹 모델의 연구[9, 13, 14]를 수행해 왔다.

본 논문은 위 연구를 통해 개발된 객체그룹 플랫폼 하에서 가상 드로잉 보드(VDB : Virtual Drawing Board) 서비스를 지원하는 시스템을 클라이언트/서버 환경으로 구축하였다. 객체그룹 모델기반의 플랫폼을 이용하는 이유는 멀티미디어 서비스 객체들이 기하급수적으로 증가하고, 서비스의 규모가 분산화 됨에 따라 단일 객체들간의 복잡한 접속과정을 피하고 객체그룹을 통해 단순한 인터페이스를 제공하도록 함이다[2]. 본 VDB 시스템의 특징은 원격 교육용 화이트 보드에서 제공하는 기능들을 가지며, 이들은 분산 객체그룹 플랫폼 상에서 지원되도록 분산 어플리케이션으로 구현되었다. 본 시스템에서 지원하는 원격교육 어플리케이션 서비스는 VDB 서버에 저장된 교육자료를 교사와 학생들이 각자의 클라이언트 시스템에서 접속, 채팅, 그래픽등의 기능들을 이용하여 양방향 학습을 지원하도록 한다.

논문의 결과로서 교육 서비스 용 분산 객체 및 객체그룹들로 구성된 VDB 시스템의 기본 구조와 객체들간의 서비스 인터페이스를 설계하고, 이를 통해 구현한 어플리케이션의 수행과정을 보였다. 구현 환경은 클라이언트/서버 환경으로써 분산 객체그룹 모델 지원이 가능하도록 CORBA(Common Object Request Broker Architecture) 미들웨어 플랫폼인 IONA사의 Orbix 2.2와 객체그룹간의 접속을 위해서 OrbixTrader 1.0과 객체그룹의 접속 및 그룹내의 객체들의 접속 보안을 지원하도록 자체 개발한 OGTG(Object Group Trader Gateway)[10]을 사용하였다.

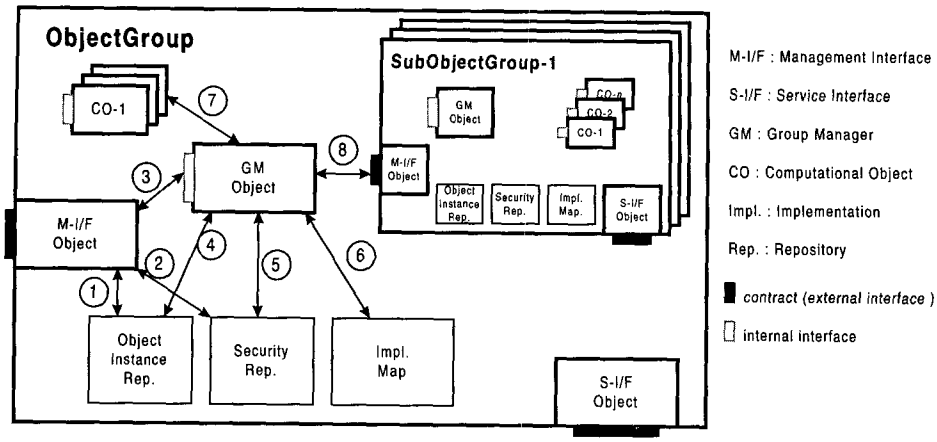
본 논문의 구성은 다음과 같다. 2장에서 분산환경을 지원하는 객체그룹 모델의 구조와 그룹 내에 구성요소들 및 그들간의 인터페이스 기능들을 정의한다. 3장은 OrbixTrader 1.0과 OGTG를 이용한 분산 객체그룹간의 상호접속과정을 기술하고, 구현되는 시스템 환경을 제시한다. 제4장에서는 객체그룹 환경 하에서 VDB 시스템의 환경과 상호작용을 위한 객체그룹 또는 객체들간의 서비스 기능들을 설계 및 구현하여 실행결과를 보였다. 마지막으로 5장에서는 결론과 앞으로의 연구방향에 대해 기술한다.

2. 객체그룹 모델

본 절에서는 우리가 그 동안 연구해 온 객체그룹 모델에서 제시된 객체그룹의 정의, 구조와 객체그룹 내의 구성요소들(components : 객체그룹 내에 속한 객체들) 및 그들간의 인터페이스 기능들에 대하여 살펴본다.

2.1 객체그룹 모델의 구조

TINA의 객체그룹 정의[6, 7]에 따라 우리는 OMG (Object Management Group) CORBA에 적합한 객체그룹 모델의 구조[9, 11]를 정립하고 구현하였다. 제안된 객체그룹의 구조와 구성요소들간의 인터페이스들은 (그림 1)과 같다. 이들 구성요소들을 세부적으로 살펴보면, 개방형 통신망 구조에서 외부와의 접속을 위해 필요한 외부 인터페이스로서 각각 관리 및 서비스 요청 시에 접근하는 관리 인터페이스 객체(M-I/F Object)와 서비스 인터페이스 객체(S-I/F Object), 객체그룹내의 관리 기능을 실질적으로 수행하는 객체그룹 관리자 객체(GM Object)가 있다. 그리고, 정보를 제공하는 객체로서 객체그룹내의 객체들에 대한 정보를 담고 있는 객체 인스턴스 레포지토리(Object Instance Repository) 객체, 객체그룹에 대한 접근 권한에 대한 정보를 담고 있는 보안 레포지토리(Security Repository) 객체 및 객체그룹 내에 생성 가능한 객체들에 대한 템플릿을 담고 있는 구현 맵(Implementation Map) 객체로 구성된다. 마지막으로 실질적인 멀티미디어 어플리케이션인 계산 객체(CO : Computational object)들이 존재한다. 여기에서 정의한 모든 객체에 대한 자세한 내용은 다음 논문들[13, 14]을 참조한다. 객체그룹은 위에서 언급한 구성



(그림 1) 객체그룹의 구조 및 구성요소간의 인터페이스

요소들 외에 서비스의 규모에 따라 한 개 이상의 서브 객체그룹을 내포할 수 있다. 또한 서브객체그룹은 구조는 상위 객체그룹과 동일한 구조를 가지며, 자신은 서브객체그룹은 가질 수 없다.

위 (그림 1)에서 객체그룹 내의 각 객체인 구성요소간의 상호작용에 따른 기능들을 살펴보면 다음과 같다. 이들 기능에 대한 오퍼레이션들의 클래스 다이어그램은 (그림 2)에서 보인다

① 관리 인터페이스 객체는 객체 인스턴스 레포지토리 객체에게 그룹 내의 객체들에 대한 정보 검색 요청을 하고, 객체 인스턴스 레포지토리 객체는 이에 대한 수행 후 결과를 반환한다.

② 관리 인터페이스 객체는 외부에서 서비스를 요청한 객체에 대한 접근 가능 여부를 보안 레포지토리 객체에게 요청하고 보안 레포지토리는 접근 가능 여부를 반환한다.

③ 관리 인터페이스 객체는 그룹관리자 객체에게 외부에서 요청한 기능 중에 보안 접근 여부와 정보 검색을 제외한, 객체의 생성, 삭제, 활성화, 비활성화와 서브객체그룹의 생성, 삭제, 활성화, 비활성화, 객체 인스턴스 레포지토리에 저장된 객체 정보에 대한 생성, 수정, 보안 레포지토리에 저장된 보안 조건의 수정, 검색 등의 요청을 하고, 그룹관리자 객체는 이에 대한 수행 후, 결과를 반환한다.

④ 그룹관리자 객체는 객체 인스턴스 레포지토리 객체에게 객체에 대한 정보를 읽기 가능, 또는 불가능으로 바꾸도록 요구하거나, 객체에 대한 정보를 찾아 줄 것을 요청한다. 이에 대해 객체 인스턴스 레포지토리 객체는 실행 결과를 반환해 준다.

⑤ 그룹관리자 객체는 보안 레포지토리 객체에게 객체에 대한 보안 조건을 생성, 삭제, 수정을 요청하고, 외부의 객체가 사용하기를 요구한 객체에 대한 보안 조건을 요청한다.

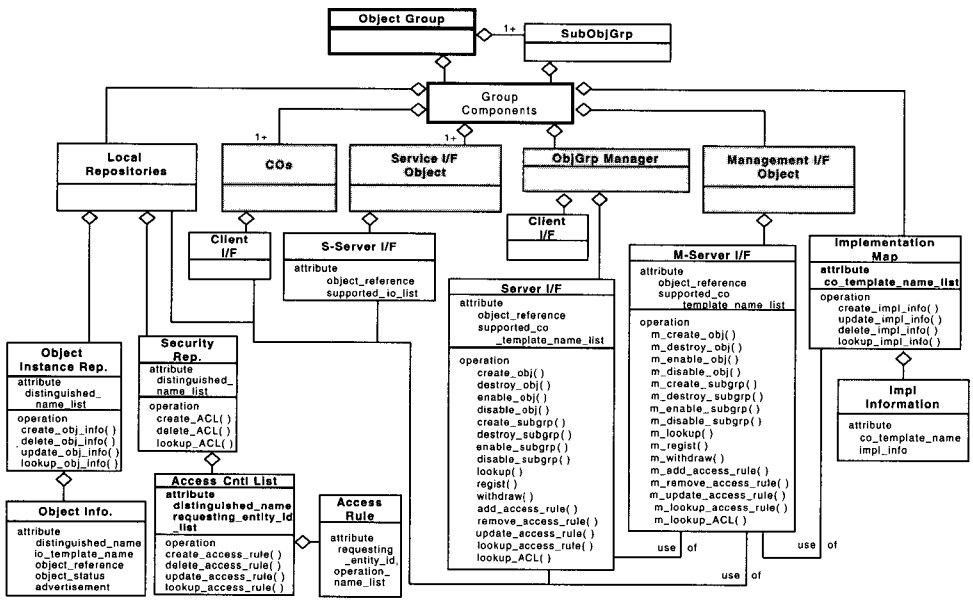
⑥ 그룹관리자 객체가 그룹 내에 객체나 서브객체그룹을 생성할 때 구현 맵 객체에게 객체의 매핑정보를 요구하게 되고, 구현 맵 객체는 이에 대한 정보를 반환한다.

⑦ 그룹관리자 객체가 객체를 생성, 삭제, 활성화, 비활성화 할 수 있음을 나타내고 있으며, 이때 그룹관리자 객체는 객체 인스턴스 레포지토리 객체에게 객체 정보를 생성, 삭제, 읽기 권한을 부여하도록 요청한다. 객체 인스턴스 레포지토리 객체는 처리된 결과를 반환해 준다.

⑧ 상위객체그룹의 그룹관리자 객체가 하위객체그룹의 관리 인터페이스 객체에게 요구를 전달하고, 하위객체그룹 내부에서 처리된 결과를 하위객체그룹의 관리 인터페이스 객체가 상위객체그룹의 그룹관리자 객체에게 반환해준다.

3. 분산 객체그룹간의 상호접속 및 시스템 환경

본 장에서는 객체그룹간의 상호접속 시, 상용 트레이더(Trader)[4, 5]와 우리가 개발한 OGTG [10, 12]를 이용하여, 개별 객체들 또는 객체그룹간의 접속 및 객체그룹 내의 객체에 대한 접속 권한을 지원한다.



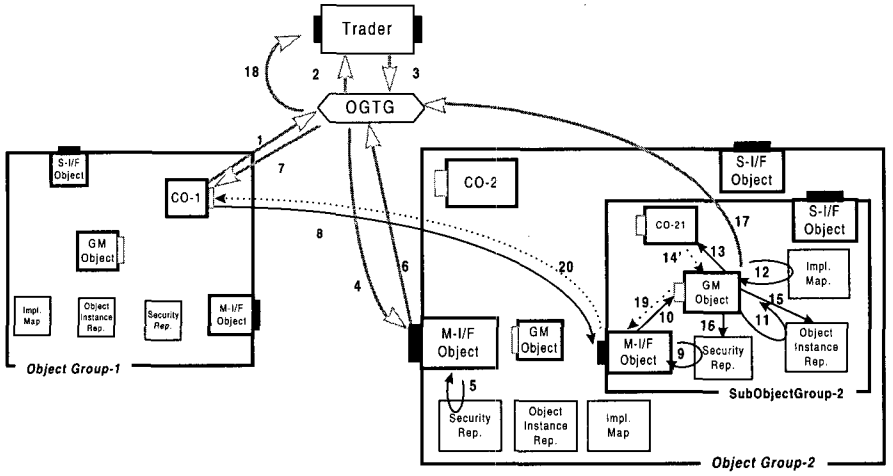
(그림 2) 구성요소들에 대한 기능 클래스 다이어그램

3.1 분산 객체그룹간의 상호접속 절차

분산 객체시스템에서는 객체들간의 접속 방법으로 속성을 이용한 트레이더를 이용하여왔다. 우리의 분산 객체그룹 플랫폼에서는 트레이딩 기능 뿐 아니라, OGTG 기능을 추가하여 객체그룹간의 접속 방법과 객체그룹들 내의 객체의 접속 시 접근 권한에 대한 문제점을 해결하도록 하였다[10, 12]. 다시 말해서, 기존의 상용 트레이더는 계층적인 객체그룹 모델(서브객체그룹을 내포한 객체그룹의 구조)을 지원하지 못하므로 객체들은 트레이더에 익스포트(export)하고 임포트(import)하는데 따른 절차와 객체그룹내의 객체 또는 서브객체그룹과 접속하려 할 때 관리상의 문제점들이 야기된다. 따라서, 이러한 문제점들을 해결하는 동시에 상용 트레이더와 병행하여 사용할 수 있도록 객체그룹과 트레이더사이에 OGTG모듈을 구현하여 연동시키므로써 분산 객체그룹들을 접속하는 방안을 제시하였다.

이 절에서는 멀티미디어 서비스를 수행하는 객체그룹의 구성요소들(객체그룹 내에 속한 객체들)에 대한 관리적인 측면만을 고려한다. 개별 객체의 서비스를 위한 접속과정에 대한 설명은 객체 지향 시스템 관점에서 많은 연구가 이루어져 왔으므로 설명은 생략하고, 객체그룹 관점에서 그들간의 관리 접속 과정[11]만을 살펴본다. 관리 접속은 객체그룹 내에 객체들의 생성, 삭제, 활성화 및 비활성화와 같은 연산(operation) 수행을 목적으로 한다. 이러한 연산의 요청은 동일한 객체그룹 또는 다른 객체그룹내의 객체에 의해서 이루어질 수 있다. 이때 동일 객체그룹내의 객체가 요청 시엔 객체들의 관리접속이 간단하나, 서로 다른 객체그룹간의 접속에서는 서로 객체그룹을 식별할 수 있어야 한다. 이때 접속을 요청하는 객체그룹은 외부와 실질적으로 접속하는 해당 객체그룹의 관리 인터페이스 객체에 대한 정보를 얻고자 트레이더와 상호작용을 한다. 하지만, 접속을 요청하는 객체와 트레이더가 직접적으로 상호 작용하는 것은 아니다. 접속을 요청하는 객체는 OGTG를 통하여 트레이더에 있는 정보를 얻어올 수 있다. 이렇게 OGTG는 트레이더와 객체그룹사이에 위치하여 그들간의 독립성을 유지시켜주면서 객체그룹 구조를 지원해준다.

객체그룹간의 상호접속 절차를 보이기 위해 객체그룹내의 멀티미디어 서비스 객체들과 관리 인터페이스 객체는 트레이더에 이미 익스포트 되어 있다고 전제하고, 객체그룹간의 접속 절차 중에서 가장 복잡한 경우로 한 객체그룹내의 연산객체가 다른 객체그룹의 서브객체그룹내의 연산객체를 생성을 나타내는 (그림 3)은 객체그룹-1(Object Group-1)내의 객체-1(CO-1)이 객체그룹-2(OG-2)의 서브객체그룹-2(SubObjectGroup-2)내에 객



(그림 3) 분산 객체그룹간의 상호접속과정(CO-21의 생성)

체-21(CO-21)을 인스턴스화시키기 위한 과정을 보인다. 참고로, 객체그룹 내에 객체들의 삭제 또는 활성화, 비활성화 연산(operation) 수행 과정은 논문[11, 14]에서 자세하게 기술하였다.

위 (그림 3)의 일련번호에 따라 접속 절차를 살펴보면 다음과 같다.

① 객체그룹-1(OG-1)안에 있는 객체-1(CO-1)이 다른 객체그룹의 서브객체그룹 내에 객체를 생성하기 위해 OGTG에게 원하는 객체그룹의 서브객체그룹에 대한 질의를 한다.

② OGTG는 요청된 질의를 트레이더에게 보내고 트레이더는 질의에 적합한 서브객체그룹을 탐색한 후, OGTG에게 서브객체그룹의 레퍼런스를 반환한다.

③ OGTG는 반환된 레퍼런스가 객체그룹-2(OG-2) 내의 서브객체그룹-2(SOG-2)에 대한 것임을 알아내어, 서브객체그룹-2(SOG-2)의 레퍼런스를 객체-1(CO-1)에게 반환하기 전에, 상위 객체그룹(OG-2)에 대한 객체-1(CO-1)의 접근 권한 검사를 대행한다.

④ ③을 하기 위해, 서브객체그룹-2(SOG-2)에 대응되어있는 상위 객체그룹-2(OG-2)의 관리 인터페이스 객체의 레퍼런스를 참조하여 접근 권한 여부를 요청한다.

⑤ 객체그룹-2(OG-2)의 관리 인터페이스 객체는 객체-1(CO-1)의 객체그룹-2(OG-2)에 대한 접근 권한 여부를 체크하여 OGTG에게 반환한다.

⑥ OGTG는 객체그룹-2에서 이루어진 접근 권한 검색 결과에 따라, 객체그룹-1(OG-1)의 객체-1(CO-1)에게 서브객체그룹-2(SOG-2)의 레퍼런스를 반환한다.

⑦ 객체그룹-1(OG-1)의 객체-1(CO-1)은 반환 받은 레퍼런스를 통해 서브객체그룹-2(SOG-2)의 관리 인터페이스 객체에게 객체 생성 요청을 한다.

⑧ 서브객체그룹-2(SOG-2)의 관리 인터페이스 객체는 서브객체그룹-2(SOG-2)의 보안 레포지토리를 통하여 요청한 객체-1(CO-1)의 서브객체그룹-2(SOG-2)에 대한 접근 권한 여부를 확인한다.

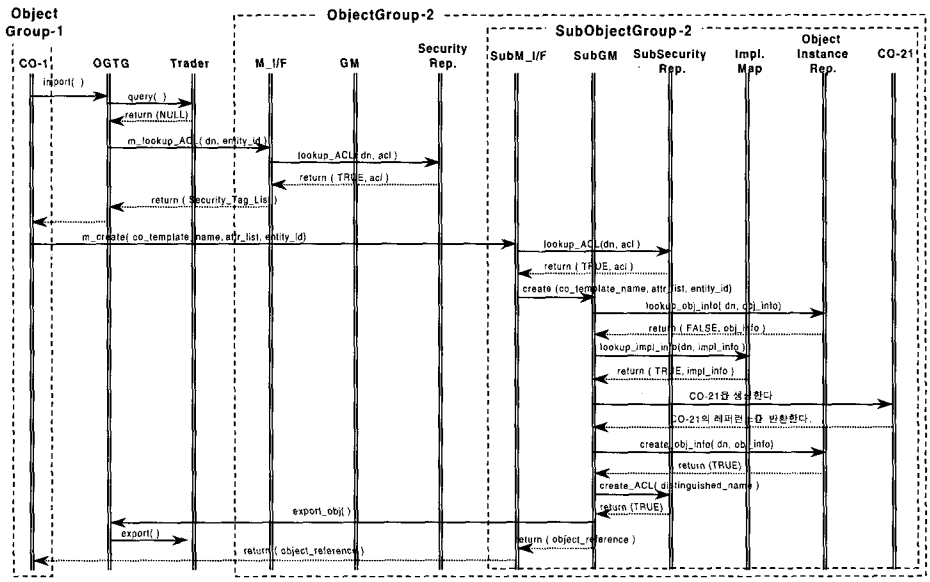
⑨ 접근 권한이 확인되면, 서브객체그룹-2(SOG-2)의 관리 인터페이스 객체는 객체 생성 요청을 서브객체그룹-2(SOG-2)의 객체그룹 관리자에게 보낸다.

⑩ 서브객체그룹-2(SOG-2)의 객체그룹 관리자는 객체-1(CO-1)이 요청한 객체의 템플릿 존재 여부를 확인하고 해당되는 객체에 대한 템플릿을 참조하여 객체-21(CO-21)을 생성(인스턴스화)한다.

⑪ 생성된 객체-21(CO-21)은 자신의 레퍼런스를 서브객체그룹-2(SOG-2)의 객체그룹관리자에게 반환하고, 서브객체그룹-2(SOG-2)의 객체그룹관리자는 이것을 서브객체그룹-2(SOG-2)의 객체 인스턴스 레포지토리에 등록한다.

⑫ 또한 객체-21(CO-21)에 대한 접근 권한 정보를 보안 레포지토리에도 등록한다.

⑬ 서브객체그룹-2(SOG-2)의 객체그룹관리자는 객체-21(CO-21)에 대한 정보를 OGTG에게 익스포트하고 OGTG는 객체-21(CO-21)의 정보를 트레이더에게 익스포트한다.



(그림 4) 분산 객체그룹간의 상호접속 오퍼레이션들의 절차도(CO-21의 생성)

⑭ 서브객체그룹-2(SOG-2)내의 객체그룹관리자는 서브객체그룹-2(SOG-2)의 관리 인터페이스 객체에게 생성된 객체-21(CO-21)에 대한 레퍼런스를 반환하고, 서브객체그룹-2 (SOG-2)의 관리 인터페이스 객체는 반환 받은 레퍼런스를 객체그룹-1(OG-1)의 객체-1 (CO-1)에게 다시 반환한다.

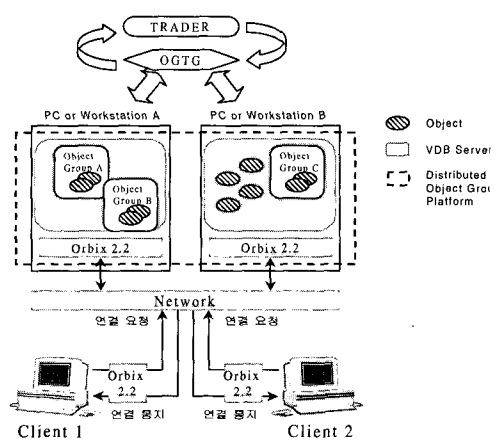
(그림 3)에 대한 객체(CO-21) 생성을 위한 구성요소들간의 수행되는 오퍼레이션들의 절차도는 (그림 4)와 같다.

4. VDB 시스템의 설계 및 구현

분산 객체그룹 기반의 VDB 시스템에 대한 시스템의 전반 구조, 설계 과정 및 구현에 대한 내용을 기술한다. VDB 시스템은 접속 및 사용자 관리를 위한 어플리케이션 서비스 객체들을 객체그룹단위로 그룹화 시켜 서버에서 실행하였으며, 클라이언트들은 서버에 접속함으로 써 접속된 또 다른 클라이언트들과 정보를 서로 공유할 수 있도록 하였다. 본 VDB 시스템의 서비스 기능들은 화이트보드의 기능들로 이를 객체화하고 분산 어플리케이션 서비스의 예로서 원격 학습과정을 보였다.

4.1 분산 객체그룹 기반의 VDB 시스템

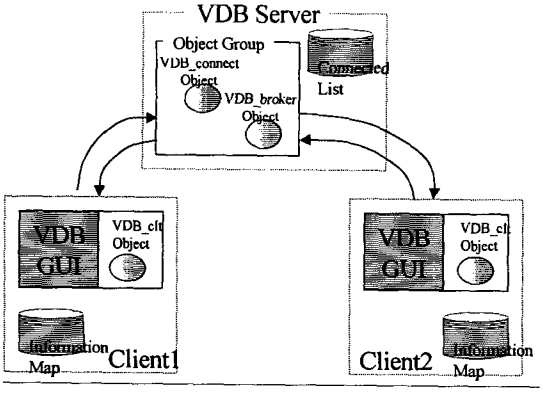
본 논문에서 구현한 분산 객체그룹기반의 VDB 시스템 환경은 아래 (그림 5)와 같다. 분산 객체그룹들 내의 객체간의 원격 교육 서비스를 위한 미들웨어로서 분산 객체 플랫폼인 CORBA 제품인 Orbix 2.2와 분산객체그룹 모델을 결합한 분산 객체그룹 플랫폼을 사용한다.



(그림 5) 분산 객체그룹 플랫폼 기반의 VDB 시스템

위 (그림 5)에서 제시하고 있는바와 같이, VDB 시스템은 서버와 클라이언트 측으로 나뉜다. 서버 측은 클라이언트들과의 연결 위한 VDB_connect 객체와 접속된 클라이언트들간의 정보 교환을 지원하는 VDB_broker 객

체로 구성되어 있으며, 이들 계산객체들은 객체그룹화 되어있다. 클라이언트 측은 서버로부터 전송되는 정보를 관리하는 VDB_clt 객체와 클라이언트들간의 정보 공유를 위한 사용자 인터페이스인 VDB GUI로 구성된다. 위 그림을 이해하기 위해서 참조로, 객체그룹의 구조는 (그림 1)에서 보인 바와 같지만, 본 장에서는 객체그룹내의 모든 구성요소들과 인터페이스는 생략하고 실제 VDB 서비스를 실행하는 계산객체들의 관점만을 고려한다. (그림 6)은 이들 서비스를 위한 계산객체(이후, 객체라고 부름)들과 이들을 클라이언트와 서버관점에서 인터페이스 과정들을 나타낸 VDB 시스템의 객체구조이다.



(그림 6) VDB 시스템의 객체 구조

4.2 VDB 시스템의 설계

VDB 시스템에서 서버스 지원을 위한 객체들의 기능 및 그들간의 접속 과정과 객체들에 대한 메소드들을 기술한다.

4.2.1 VDB 서비스 객체들과 그들의 기능들

VDB 시스템은 VDB_connect, VDB_broker, VDB_clt, VDB GUI 그리고 Information Map, Connected_List 객체들로 구성되어 있다. 각 객체들의 기능들을 살펴보면 다음과 같다.

- VDB_connect 객체

현재 VDB 시스템에 접속한 클라이언트들의 연결과 해지에 관한 정보를 Connected_List 객체를 통해 관리한다.

- VDB_broker 객체

'접속된 모든 클라이언트로부터 생성된 Event를 VDB 시스템에 접속하고 있는 다른 모든 클라이언트에게 전송한다.

- VDB_clt 객체

VDB_broker 객체로부터 전송된 정보를 토대로 클라이언트의 Information Map을 관리한다.

- VDB GUI 객체

VDB 시스템에 접속한 클라이언트들의 사용자 인터페이스를 지원한다.

- Information Map 객체

VDB 시스템에 접속한 클라이언트의 사용자 인터페이스에 의해 보이는 정보를 저장한다.

- Connected_List 객체

VDB 시스템에 접속되어 있는 사용자들의 주소 정보를 보여준다.

4.2.2 접속 과정

- 연결 과정

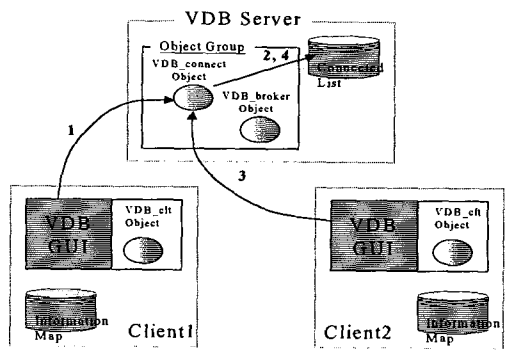
객체들간의 접속 과정은 서버와 클라이언트들의 연결, 클라이언트 상호간의 정보 교환, 클라이언트들의 연결 해지 과정들로 나눈다. VDB 서버와 클라이언트간의 연결과정은 다음과 같이 이루어진다(그림 7).

① 클라이언트1이 VDB_connect 객체에게 연결을 요청한다.

② VDB_connect 객체가 Connected_List에 클라이언트1의 주소를 추가한다.

③ 클라이언트2가 VDB_connect 객체에게 연결을 요청한다.

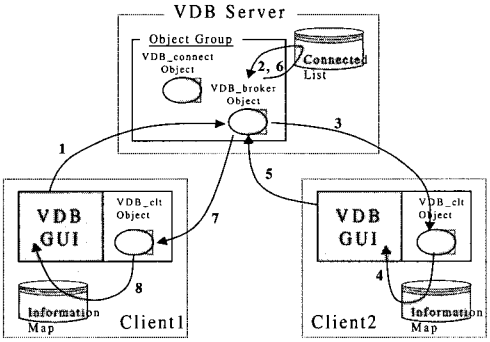
④ VDB_connect 객체가 Connected_List에 클라이언트2의 주소를 추가한다.



(그림 7) VDB 서버와 클라이언트들간의 연결 과정

● 클라이언트들간의 상호정보 교환 과정

VDB 서버에 연결되어 있는 클라이언트들 상호간에 정보를 교환하는 과정은 (그림 8)과 같이 나타낸다.



(그림 8) VDB 서버를 이용한 클라이언트들간의 상호 정보교환과정

<클라이언트1이 자신의 화면에 정보를 입력하는 경우>

① 클라이언트1의 VDB GUI가 VDB_broker 객체에게 자신의 변경된 정보를 전송한다.

② VDB_broker 객체는 Connected_List에서 현재 접속되어있는 모든 클라이언트들의 정보를 얻는다.

③ 접속된 모든 클라이언트의 VDB_clt 객체에게 변경된 정보를 보낸다.(여기에서는 클라이언 트2에게만 보냄)

④ 클라이언트2의 VDB_clt 객체는 자신의 Information Map과 VDB GUI 화면을 갱신한다.

클라이언트2가 자신의 화면에 정보를 입력한 경우도 마찬가지의 절차를 수행한다(⑤, ⑥, ⑦, ⑧ 참조).

● 연결 해지 과정

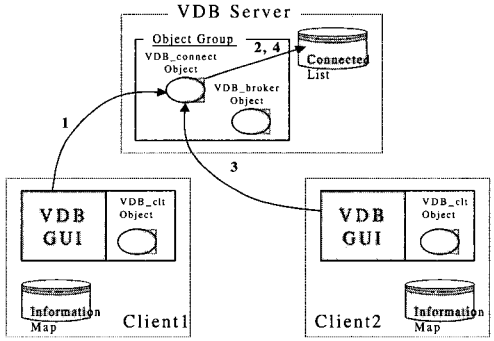
클라이언트들이 VDB 서버와 연결 상태를 해지하는 과정은 (그림 9)에서 나타낸다.

① 클라이언트1이 VDB_connect 객체에게 연결 해지를 요청한다.

② VDB_connect 객체가 Connected_List 객체를 통해 클라이언트1의 주소를 삭제한다.

③ 클라이언트2가 VDB_connect 객체에게 연결 해지를 요청한다.

④ VDB_connect 객체가 Connected_List 객체를 통해 클라이언트2의 주소를 삭제한다.



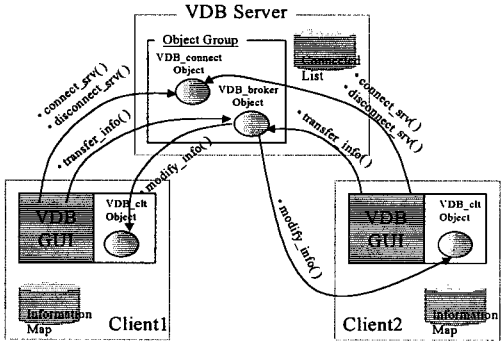
(그림 9) VDB 서버와 클라이언트의 연결해지 과정

4.2.3 객체들에 대한 메소드들과 이들간의 상호작용

객체들의 기능과 접속 과정을 통하여 추출한 VDB 시스템에서 지원되는 객체들의 메소드들은 다음과 같다.

- DB_connect 객체 : connect_srv(), disconnect_srv()
- VDB_broker 객체 : transfer_info()
- VDB_clt 객체 : modify_info()
- VDB_GUI : paint_Point(), aint_Line(), aint_ctangle(), paint_ llipse(), nput_Text()

(그림 10)은 위에서 기술된 메소드들에 의한 VDB 서버와 클라이언트 객체들간의 상호작용을 나타낸다.

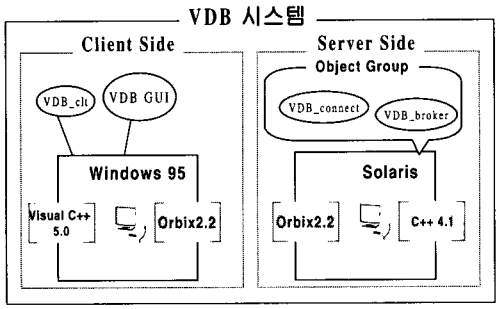


(그림 10) 메소드들에 의한 VDB 서버와 클라이언트 객체들의 상호작용

4.3 VDB 시스템의 구현

본 절에서는 VDB 시스템의 물리적 환경 및 구현 모듈 그리고, 구현된 실행 화면에 대해서 설명한다. 클라이언트와 서버간의 상호작용을 위한 VDB_connect, VDB_broker, VDB_clt 객체들을 CORBA 객체로 만들기 위해서, 이들간의 인터페이스 과정을 IDL(Interface

Definition Language)을 이용하여 작성한다. 객체들간의 인터페이스를 IDL로 작성한 프로그램을 컴파일 하여 클라이언트용 stub와 서버용 skeleton 프로그램을 생성한다. skeleton 프로그램과 VDB_connect와 VDB_broker 모듈들은 UNIX상에서 solaris C++ 4.1로, stub 프로그램과 VDB_clt 모듈은 Windows 95상에서 Visual C++ 5.0으로 각각 작성하고, 이를 컴파일과 링크과정을 거쳐 서버용 실행 프로그램과 클라이언트용 실행 프로그램으로 구현한다. (그림 11)은 구현된 VDB 환경을 보인다. 실행 결과 화면으로는 VDB GUI의 초기화면과 클라이언트와 서버의 접속 화면, 두 클라이언트간의 정보 교환, 서버와의 연결 해지 화면들을 보였다.



(그림 11) VDB 시스템의 구현 환경

4.3.1 구현 모듈

- VDB_connect

VDB_connect 객체는 클라이언트가 서버와 접속을 위해 사용하는 객체이다. 오퍼레이션으로 connect_srv와 disconnect_srv를 갖는다. connect_srv 오퍼레이션은 클라이언트가 서버에 접속하기 위하여 사용한다. 여기에서 사용되는 매개 변수로는 클라이언트의 주소 정보이다. disconnect_srv 오퍼레이션은 클라이언트가 서버와의 접속을 해지하기 위해 사용되며, 이 오퍼레이션의 매개 변수도 역시 클라이언트의 주소 정보가 된다.

```
interface VDB_connect {
boolean connect_srv(in string clt_addr);
boolean disconnect_srv(in string clt_addr);
}
```

- VDB_broker

VDB_broker 객체는 클라이언트가 변경된 자신의 정보를 다른 클라이언트들에게 알리기 위해서 사용하는 객체로 transfer_info 오퍼레이션이 기술된다. transfer_info 오퍼레이션은 클라이언트가 서버에게 변경된 자신의 정보를 전송키 위해 사용하는 오퍼레이션이다. 매개 변수로는 정보를 전송하는 클라이언트의 주소 정보와 변경된 공유정보(Shared_Info map)가 정의된다. 이러한 정보를 전송한 후, 클라이언트는 다른 작업을 계속하여야 하기 때문에 오퍼레이션만 실행만 시키고 제어는 다른 작업을 위해 사용할 수 있도록 다시 가져올 수 있도록 oneway 오퍼레이션으로 지정하였다.

```
typedef sequence <char> Shared_Info;
interface VDB_broker {
oneway void transfer_info(in string clt_addr,
in Shared_Info map);
}
```

- VDB_clt

VDB_clt 객체는 서버 측으로부터 클라이언트에게 변경된 정보를 전송하기 위해 사용되는 객체이다. 오퍼레이션으로는 modify_info를 가지고 있다. modify_info 오퍼레이션은 변경된 정보의 전송을 위해 사용된다. VDB_broker 객체의 transfer_info 오퍼레이션과 같이 정보를 전송한 후, 클라이언트가 다른 작업을 계속하여야 하기 때문에 오퍼레이션은 실행만 시키고 제어는 다시 자신이 가지고 올 수 있도록 oneway 오퍼레이션으로 지정하였다.

```
typedef sequence <char> Shared_Info;
interface VDB_clt {
oneway void modify_info(in Shared_Info map);
}
```

4.3.2 실행과정 및 수행결과 화면

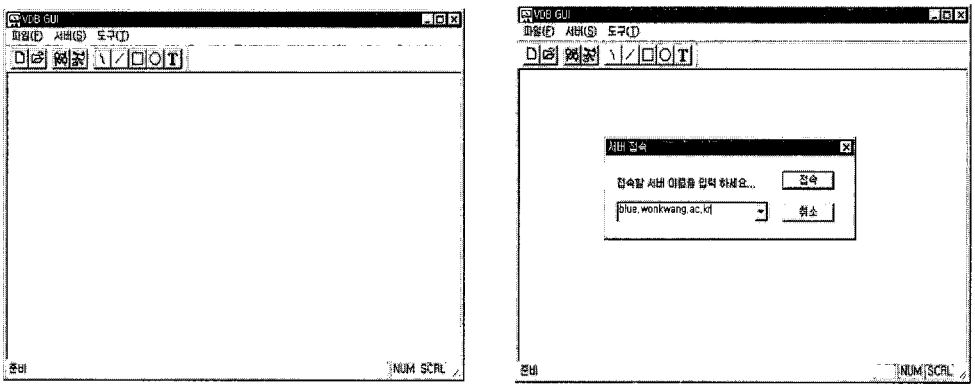
본 논문에서 구현한 VDB 시스템의 실행 환경은 구현 환경과 동일하다. VDB 시스템은 다음과 같은 기본 과정을 통하여 동작시킬 수 있으며, 여기에서 보이는 기능을 기본으로 하여 확장 가능하다.

① 클라이언트가 VDB GUI를 실행시킨다.

② VDB 서버에 접속을 요청한다.

③ 접속된 클라이언트들은 자신의 VDB GUI를 이용하여 다른 클라이언트들과 정보를 교환한다.

④ 서버와의 접속을 해지한다.



(그림 12) 클라이언트의 초기화면과 VDB 서버와의 연결 과정

● VDB GUI의 초기화면

클라이언트에서는 VDB 시스템의 사용을 위해서 VDB GUI를 사용하여야 한다. VDB 환경에서 GUI는 보다 편리한 사용을 위해 윈도우 기반의 인터페이스를 제공한다. 기본 기능으로는 기본 메뉴로는 파일, 서버, 도구 메뉴가 있다. 파일 메뉴에는 화면갱신, 파일열기, 인쇄, 종료가 있으며, 서버 메뉴에는 접속하기, 접속끊기가 있다. 도구 메뉴에는 점 그리기, 선 그리기, 사각형 그리기, 원그리기, 문자열 입력하기가 있다. (그림 12)의 좌측 그림은 클라이언트에서 VDB GUI를 실행했을 때의 초기 화면을 보이고 있다.

● 클라이언트와 VDB 서버와의 연결 과정

클라이언트가 VDB 서버와 연결을 하고자 할 때, 서버 메뉴에서 접속하기를 선택하면, 서버 접속을 위한 다이얼로그 박스가 나타난다. 다이얼로그 박스의 입력 상자에 접속을 원하는 서버의 주소를 입력함으로써 접속을 할 수 있다. (그림 12)의 우측화면은 VDB 서버와의 연결과정을 화면으로 보인다.

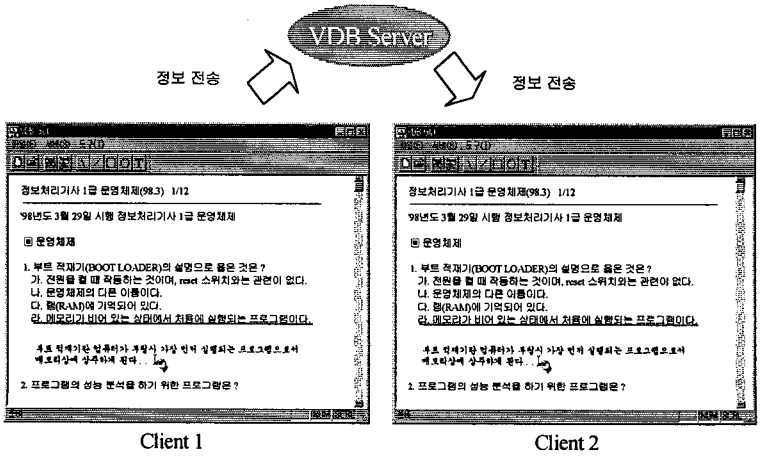
● 클라이언트간의 정보 교환

클라이언트 1과 2가 VDB 서버를 접속하여 정보를 교환하는 과정은 다음과 같다.

① 클라이언트 1과 클라이언트 2가 각각의 VDB GUI를 실행한다.

② 클라이언트 1과 클라이언트 2가 VDB 서버에 접속한다(그림 12).

③ 클라이언트 1이 도구 메뉴 중 문자열 입력 기능



(그림 13) VDB 서버를 이용한 클라이언트1과 2의 화면상에 공유된 정보

을 이용하여 자신의 VDB GUI 환경에서 에 디터를 통해 학습내용을 입력 또는 해당 스캔된 자료를 .pdf 화일로 작성한다.

④ 입력된 정보는 VDB 서버에 저장되고, 서버를 통해 클라이언트2에 전송된다(이 때 학습내용은 클라이언트1과 2의 두 화면에 보여진다).

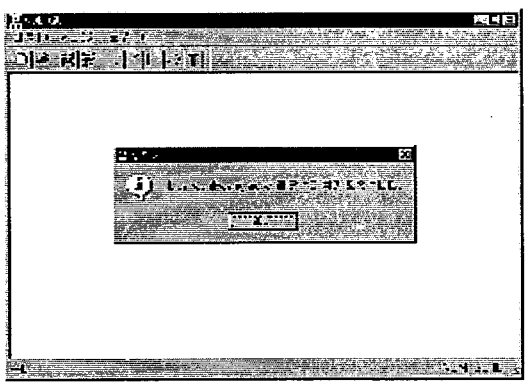
⑤ 클라이언트1이 화면에 보여진 학습내용 위에 텍스트나 그래픽으로 작성된 내용은 VDB 서버를 통해 클라이언트2의 GUI 화면에 보여진다.

⑥ 학습이 끝나면, 클라이언트1과 클라이언트2가 VDB 서버와 접속을 끊는다(그림 14).

다음 (그림 13)은 ③~⑥의 과정을 통합하여 보인다.

- VDB 서버와의 연결 해지

클라이언트가 VDB 서버와 연결을 해지 하고자 할 때, 서버 메뉴에서 접속끊기를 선택하면, 연결 해지를 알리는 메시지 창이 나타난다. (그림 14)는 VDB 서버와의 접속 해지 메시지 창을 클라이언트 화면을 통해 보인다.



(그림 14) VDB 서버 접속 해지 메시지 화면

5. 결 론

분산 컴퓨팅 환경은 다양한 멀티미디어 서비스의 질적, 양적 요구사항들을 수용할 수 있도록 제안하고 있는 TINA-C와 OMG CORBA의 권고안에 따라 객체지향형 미들웨어 기반의 개방형 정보통신망 구조가 연구되고 있다. 그러나 멀티미디어 서비스 어플리케이션의 규모가 점점 커지고 분산화 됨에 따라 분산 객체들간의 서비스 및 관리 인터페이스가 매우 복잡해지고 있다. 이러한 단점을 해결하기 위해서 새로운 객체그룹 모델의 제안과 객체그룹 하에서 도입될 수 있는 객체 관리 및 서비스를 위한 요구사항들이 필요하다. 따라서 우리는 기존 객체들간의 복잡한 접속과정을 감소시킬 수 있고, 객체들을 그룹화 하여 편리하게 관리할 수 있도록 모델링한 분산 객체그룹 플랫폼을 개발하였다.

본 논문에서는 이러한 분산 객체그룹 플랫폼을 기반으로 하여 VDB 시스템을 설계 및 구현하였다. 본 VDB 시스템의 특징은 원격 교육용 화이트 보드에서 제공하는 기능들을 가지며, 이들은 분산 객체그룹 플랫폼 상에서 지원되도록 분산 어플리케이션으로 구현되었다. 이러한 객체지향 미들웨어의 사용은 이 기종 시스템들간에 객체들의 상호동작(interoperability)이 가능하고, 그들간에 동적 서비스 지원으로 네트워크 트래픽을 감소시킬 수 있는 장점을 가진다.

본 시스템에서 지원하는 원격교육 어플리케이션 서비스는 VDB 서버에 저장된 교육자료를 각각의 클라이언트 시스템에서 접속, 채팅, 그래픽등의 기능들을 이용하여 양방향 학습을 지원하도록 한다. 이를 위해, 본 논문에서는 분산 객체 및 객체그룹들로 구성된 VDB 시스템 환경을 제시하고, VDB 서버와 클라이언트들 내의 객체들간의 오퍼레이션 서비스를 위한 인터페이스들을 설계하였다. 마지막으로 구현한 분산 어플리케이션 서비스의 수행절차를 보였다.

본 VDB 시스템은 분산 객체그룹 플랫폼 상에서 오퍼레이션 접속 방법에 의한 서비스만을 지원하고 있다. 앞으로 이러한 기반을 확장하여 VDB 오퍼레이션 기능 뿐 아니라 오디오나 비디오와 같은 연속 매체(continuos media)의 서비스 지원을 위한 스트림(stream) 객체의 구현과 접속방법 및 QoS 지원 방안을 연구해 나갈 계획이다.

참 고 문 헌

[1] Bersia, Bosco, Monione, Moiso, and Spinolo, "A CASE environment for TINA-oriented application," CSELT, 1994.

[2] Silvano Maffeis, "The Object Group Design Pattern," Dept. of Computer Science, Cornell Univ. 1996. http://cs-tr.cs.cornell.edu:80/Dienst/UI/1.0/Display/ncstrl.cornell/Technical Report 96-1570.

[3] Pier Giorgio Bosco and Corrado Moiso, "A Distributed Processing Model for Telecommuni-

cations Service Management," The Proceedings of DSOM '95, 1995.

[4] OMG, "OMG RFP5 Submission : Trading Object Service," 1996.

[5] OMG, "CORBA Services : Common Object Service Specification," 1997.

[6] TINA-C, "TINA ODL-Manual Version. 2.3," 1996. 7.

[7] 신영석, 오현주, 고병도, 김재근, "차세대 개방타입 정보통신망 구조인 TINA 연구", 한국전자통신연구원 주간기술 동향, 744/5호, 1996.

[8] 신영석, 오현주, "이 기종 분산처리환경에서 연결관리 객체의 정보공유", 한국통신학 회논문지 제22권 4호, 1997.

[9] 고창록, 신영석, 김명희, 주수종, "개방형 분산 시스템에서 객체그룹 모델링에 관한 연구", 한국정보과학회 추계학술발표지, 24권 2호, pp.27-30, 1997. 10.

[10] 김명희, 신경민, 이재완, 주수종, "분산처리환경에서 객체그룹간 연산객체들의 상호접속", 한국정보과학회 논문지, 제25권 8호, pp.815-824, 1998. 8.

[11] 주수종, "분산처리환경에서 객체그룹 모델링 및 성능분석에 관한 연구", 최종연구보고서, ETRI, 1997. 11.

[12] 김명희, 신영석, 정창원, 주수종, "The Interconnection Modelling of Object Groups Using Trader and OGTG in Distributed Processing Systems," IASTED International Conference of Simulation and Modelling, Pittsburgh, U.S.A., pp.405-409, 1998. 4. 17.

[13] 이승용, 정창원, 신영석 주수종, "개발형 분산 환경에서 객체그룹 모델의 설계", 한국 통신학회 논문지, 제23권 9호, pp.2258-2270, 1998. 9.

[14] 김명희, 신영석, 정창원, 신경민, 주수종, "분산처리환경에서 멀티미디어 서비스를 위한 객 체그룹 모델의 설계 및 구현", 한국정보처리학회 학술지, *CD-ROM file://F\/Proceeding/non13/non12.html*, 1998. 4.

[15] 이원중, 안길수, 주수종, "웹 환경에서 확장된 객체지향형 미들웨어(CORBA/JAVA)를 이용한 멀티미디어 정보 검색시스템 구현", 한국정보처리학회논문지, 제5권 7호, pp.1847-1854, 1997. 7.

유 경 택

e-mail : rhyukt@kdnms.kdc.ac.kr, rhyukt@nownuri.net

1988년 원광대학교 전자계산공학과 졸업(공학사)

1990년 광운대학교 대학원 전산기공학과 졸업(공학석사)

1998년 원광대학교 대학원 컴퓨터공학과 박사과정

1995년~현재 극동정보대학 전산정보처리과 조교수

관심분야 : 분산컴퓨팅, 멀티미디어 문서처리, 하이퍼미디어 시스템

이 현 철

e-mail : snowman1@netian.com

1997년 원광대학교 컴퓨터공학과 졸업(공학사)

1998년~현재 원광대학교 대학원 컴퓨터공학과 석사과정

관심분야 : 분산 컴퓨팅, 스트리밍 기술, 멀티미디어 데이터베이스

주 수 종

e-mail : scjoo@wonkwang.ac.kr

1986년 원광대학교 전자계산공학과 졸업(공학사)

1988년 중앙대학교 대학원 컴퓨터공학과 졸업(공학석사)

1992년 중앙대학교 대학원 컴퓨터공학과 졸업(공학박사)

1993년~1994년 미국 Univ of Massachusetts at Amherst 전기 및 컴퓨터공학과 Post-Doc.

1990년~현재 원광대학교 컴퓨터공학과 부교수

관심분야 : 분산 실시간 컴퓨팅, 분산객체모델, 시스템 최적화, 멀티미디어 데이터베이스