가상 기업의 통합 데이터 환경을 위한 데이터베이스 브로커 시스템의 설계 및 구현

윤 선 희† · 정 진 욱††

요 약

네트워크 컴퓨팅의 급속한 발전과 기업의 인터넷/익스트라넷의 일반적인 사용으로 미래의 비즈니스 환경은 기업의 이익 및 효율성을 최대화하기 위해 기업 통합(Enterprise Integration) 및 가상 기업(Virtual Enterprise)가 실현 될 것으로 예상된다. 기업간의 정보 공유를 기반으로 하는 가상 기업을 운영하기 위해서는 가상 기업의 정보 전달 수단과 관련된 작업 프로세스, 가상 기업에 참여하는 구성원 및 CALS의 구현을 기반으로 하는 가상 기업의 지원 환경으로 이루어진다. 가상 기업의 환경으로 제공되는 CALS 구현을 위한 통합 데이터 환경을 지원하는 시스템은 각 지역 데이터의 자치성을 보장하며 사용자에게 단일의 전역적인 뷰를 제공하기 위해 네트워크로 연결된 기업간의 이질적인 데이터베이스를 투명하게 엑세스할 수 있도록 제공되어야 한다. 본 논문에서는 가상기업 지원 환경으로 제공되는 통합데이터 환경을 위하여 각 기업에 존재하는 기존의 시스템을 유지하면서 이질적인 데이터베이스들을 투명하게 엑세스할 수 있는 방법으로 웹환경에서 Java/CORBA 기술과 관계형 데이터베이스, 객체지향형 데이터베이스와 파일 정보를 수용하기 위한 객체 질의 언어를 사용하는 데이터베이스 브로커 시스템을 설계 및 구현한다.

Design and Implementation of Database Broker System for Integrated Data Environment of Virtual Enterprises

Sun-Hee Yoon† · Jin-Wook Chung††

ABSTRACT

In recent days network computing technologies have been developed rapidly and the extended use of Internet applications for enterprises such as intranet/extarnet in and between enterprises has been increased enormously. Therefore the business in the future will be executed by virtual enterprise. Virtual enterprises which is based on information sharing between enterprises are composed of work processes related to information exchange between virtual enterprises, the team members who are representatives of the organizations that are participated in the actual business of virtual enterprises, and environment that are provided by supporting CALS(Continuous Acquisition and Life cycle Support or Commerce At Light Speed). Supporting system of IDE(Integrated Data Environment) for CALS implementation that is provided as an environment of virtual enterprises has to ensure the autonomies of local data and to provide the accessibility of heterogeneous database of enterprises on network transparently for giving user a single global view of data.

This paper introduce the design and implementation of the database broker system that can be accessed data transparently by the users of participated enterprises in the integrated data environment supporting virtual enterprises. The system uses Java/CORBA technology in Web environment and Object Query Language (OQL) to process the queries of relational database system, object-oriented database system, and file information.

† 정 회 원 : 한국전자통신연구원 컴퓨터·소프트웨어기술연구소, CALS 연구실 선임연구원
†† 종신회원 : 성균관대학교 전기·전자 및 컴퓨터공학부 교수
논문접수 : 1998년 9월 30일, 심사완료 : 1998년 12월 30일

1. 서 론

네트워크 컴퓨팅의 급속한 발전과 기업의 인터넷/익스트라넷의 일반적인 사용으로 미래의 비즈니스 환경은 기업의 이익 및 효율성을 최대화하기 위해 기업 통합(Enterprise Integration) 및 가상 기업(Virtual Enterprise)이 실현 될 것으로 예상된다. 기업간의 정보 공유를 기반으로 하는 가상 기업을 운영하기 위해서는 가상 기업의 정보 전달 수단과 관련된 작업 프로세스, 가상 기업에 참여하는 구성원 및 CALS(Continuous Acquisition and Life cycle Support or Commerce At Light Speed)의 구현을 기반으로 하는 가상 기업의 지원 환경으로 이루어진다.

가상 기업의 환경으로 제공되며 기업간의 정보 공유를 기반으로 하는 CALS 구현을 위한 통합 데이터 환경을 지원하는 시스템은 각 기업에 물리적으로 분산된 이질의 데이터베이스 시스템들을 단일의 전역적인 뷰를 제공하기 위하여 논리적으로 통합하여야 한다. 이러한 시스템을 구축하기 위한 방법으로는 하향식 접근 방식과 상향식 접근방식이 있다. 하향식 접근 방식은 시스템을 새로 구축하는 방식으로 전역(global) 스키마가 구축되며 각 지역 스키마와 지역 데이터베이스 시스템에서 제공되는 역할이 결정되는 것으로 기존의 사용중인 시스템은 고려하지 않는다. 반면 상향식 접근 방식은 기존 시스템의 자치성을 유지하며 기존의 지역 데이터베이스의 스키마들 중에서 외부 스키마를 추출하여 전역 스키마를 생성하는 방식으로 다중 데이터베이스(multidatabase) 시스템이 대표적인 예이다[11, 15].

가상 기업의 데이터를 공유하기 위한 지원 환경의 기본 개념이 시스템을 새로 구축하는 것이 아니라 기존 시스템을 가능한 사용하는 것이기 때문에 CALS 구현을 위한 통합 데이터 환경을 지원하기 위한 시스템은 다중 데이터베이스 시스템의 구축을 적용할 수 있다. 그러나 다중 데이터베이스 시스템은 각 기업에 분산된 데이터베이스 시스템의 스키마를 통합하여 주어야 하는 제약이 있으며 사용자의 요구 및 기업의 데이터베이스 변환에 의한 동적 스키마 통합이나 스키마 통합시에 발생할 수 있는 의미 충돌(semantic conflict)를 해소할 수 있기 위한 자동 스키마 통합이 용이하지 않다.

본 논문의 목적은 가상 기업의 데이터 공유 지원 환경을 위한 통합 데이터 환경을 지원할 수 있는 시스템으로써 데이터베이스 브로커 시스템을 제안한다. 데이터베이스 브로커 시스템은 데이터베이스 전위기(front-end)로써 데이터베이스 관리 시스템을 호출하는 응용 소프트웨어와 데이터베이스 관리기 사이에서 인터페이스 계층의 역할을 담당한다. 본 논문에서 제안된 데이터베이스 브로커 시스템은 각 기업에 존재하는 기존의 시스템을 유지하면서 다중 데이터베이스 시스템의 단점인 전역 스키마의 통합을 이루지 않으며, 이질적으로 분산된 데이터베이스들을 투명하게 접근(access)할 수 있는 장점을 가진다. 제안된 데이터베이스 브로커 시스템은 인터넷 환경에서 Java를 사용하여 사용자가 데이터베이스 접속에 의해 질의 요청시 일관성 있는 세션 관리가 유지되게 하며, 기업간의 이질 플랫폼상의 상호 운용성 및 위치 투명성을 제공하기 위해 통신 미들웨어로써 CORBA(Common Object Request Broker Architecture)를 사용한다. 이러한 데이터베이스 브로커 시스템은 사용자에 의해 요구된 데이터와 데이터 모델 및 플랫폼에 있어서 기업간의 정보 공유가 효율적으로 이루어지도록 하며 지역 데이터들의 자치성 보장 및 정보 검색에 있어서 고도의 투명성을 제공한다.

2. 통합 데이터 환경 지원 시스템의 연구 현황 및 분석

가상 기업의 지원 환경인 CALS의 통합 데이터 환경을 제공하기 위해 구축된 대표적인 프로젝트로는 미국방부에서 개발한 JCALS(Joint CALS)의 GDMS (Global Data Management System)과 일본 NCALS (Nippon CALS)에서 구현한 수평적 분산 데이터베이스 시스템이 있다.

2.1 연구 현황

(1) JCALS GDMS(Global Data Management System)

JCALS는 미 국방부의 육□해□공 3군, 해병대 및 연방 후방 지원청이 결합하여 국방부 산하의 CALS환경과 기능을 확립하고 무기체계 시스템의 조달 및 후방 지원을 위하여 구축되었다 [1,2].

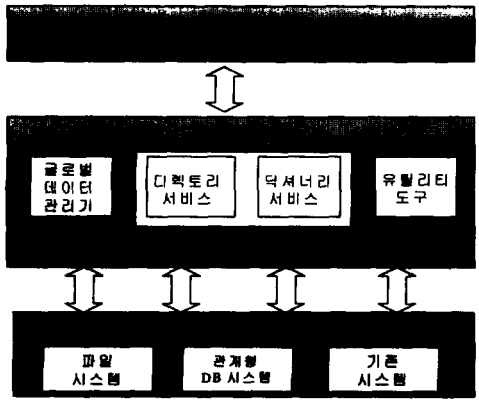
JCALS의 GDMS는 국방 환경에서 다양한 소프트웨어 응용을 위한 기능들을 지원하기 위한 서비스 중개자(broker)로서의 역할을 하며, 물리적으로 분산된 데이터들을 논리적으로 통합하기 위한 수단을 제공한다

JCALS의 GDMS는 위치에 상관없이 분산 데이터 환경에서 정보를 관리하고 접근하기 위해 요구되는 서비스들을 제공하기 위한 시스템이다.

JCALS에서 미들웨어로 제공되는 GDMS는 분산 데이터베이스 시스템의 하향식 접근 방식을 사용하여 중앙 집중적 통제에 의해 관계형 데이터베이스 관리 시스템을 기반으로 하는 동질의 통합 데이터베이스 환경을 지원하며 각 기능 요소는 밀접하게 결합된 형태(tightly-coupled)로 상호 작용한다.

GDMS의 구조 (그림 1) 및 서비스 기능은 다음과 같다.

- 사용자의 요청에 의한 SQL 기반의 질의 해석 기능
- 사용자의 접근 권한 및 데이터 접근 제어를 관리하는 기능
- 전역 질의를 지역 질의로 변환하는 기능
- 지역 질의를 전역 질의로 변환하여 통합하는 기능



(그림 1) GDMS 구조[1]
(Fig. 1) GDMS Architecture[1]

전역 데이터 관리를 위한 전역 데이터 처리기 기능은 네트워크 관리 및 데이터의 분산 및 복사에 대한 관리를 담당한다. 전역 사전/디렉토리(global dictionary /directory) 서비스 기능은 검색 엔진과 컴파일된 형태의 뷰 트랜잭션을 위한 데이터의 위치 정보를 관리하며 매핑시켜 주는 역할을 담당한다. 지역 처리기 서버는 지역에 분산된 이질의 시스템에 의해 정의된 데이터들을 접근하고 검색하기 위한 데이터 매핑 기능을 제공한다. 또한 지역간의 데이터들의 인터페이스를 위한 변환 기능을 제공한다. 전역 저장소에서 관리하는 데이터의 형태는 관계형 데이터베이스 시스템의 데이터 및 파일 정보를 포함한다.

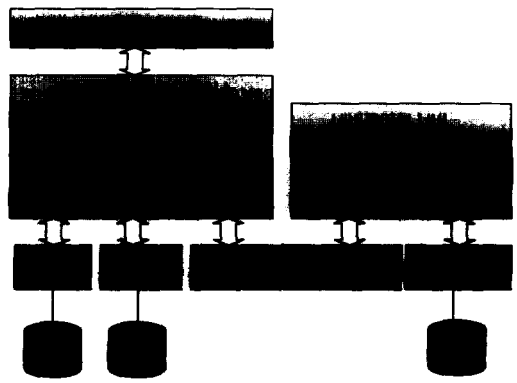
(2) NCALS의 수평적 분산 데이터베이스 시스템

미국이 국방부의 주도하에 군 무기 시스템의 전산화를 위한 방위 산업 분야를 중심으로 CALS 프로젝트가 추진된 것과는 달리 일본은 민간 기업이 주도적으로 추진하면서 상호 관련 기업이 컨소시엄 형태로 참여하는 정책을 추진하고 있다[8, 9].

NCALS에서 추진하고 있는 통합 데이터 환경을 지원하는 수평적 분산 데이터베이스 시스템은 관계형 데이터베이스 관리 시스템을 기반으로 하며, 통합 데이터 환경을 제공하기 위해 기업간의 중복된 교환 색인(index) 데이터베이스를 사용하여 정보를 항해(navigation)하기 위한 기능을 제공한다.

NCALS의 수평적 분산 데이터베이스 시스템에서 사용된 구조 (그림 2) 및 기능은 다음과 같다.

- 지역 정보를 질의 및 관리하기 위한 기능
- 색인 정보를 관리하기 위한 기능
- 문서 파일을 전송하기 위한 기능
- 응용 시스템과의 인터페이스를 제공하는 데이터베이스 접근 기능
- 사용자 권한에 의한 접근 제어 기능



(그림 2) NCALS의 수평적 분산 데이터베이스 시스템 구조[9]
(Fig. 2) NCALS Horizontally Distributed Database System Architecture[9]

NCALS의 수평적 분산 데이터베이스 시스템의 기능은 지역 서버 기능 및 항해 서버 기능으로 구성된다. 지역 서버에서 제공하는 기능은 외부 응용 시스템과의 인터페이스를 담당하는 통합 데이터베이스 접근 인터페이스 기능, 요청된 문서를 위한 지역 정보를 알기 위해 항해 서버에 요청하는 지역 정보 질의 기능,

지역의 데이터 사전/디렉토리(LDD/D), 지역 정보 데이터베이스 등과 같은 데이터베이스를 참조하거나 갱신하기 위한 지역 정보 관리 기능, 특정 지역들간의 문서를 전송하는 문서 전송 기능들로 구성된다. 항해 서버에서 제공하는 기능은 지역들간의 정보 교환을 위한 색인 정보를 관리하는 색인 정보 관리 기능, 사용자로부터 요청된 질의 및 사용자 권한을 관리하는 접근 제어 기능, 사용자의 요청에 의한 색인 정보의 검색에 실패했을 경우, 다른 지역에 검색을 요청하는 기능 등으로 구성된다.

2.2 GDMS와 수평적 분산 데이터베이스 시스템의 분석

JCALS에서 구현한 GDMS는 하향식 접근 방식으로써 중앙 집중적 통제에 의한 수직적 분산 데이터베이스 시스템 형태의 통합 데이터 베이스 환경을 지원한다[1,2]. 이러한 통합 데이터베이스 환경을 지원하는 것은 국방부와 같은 통제권이 부여된 정부 기관에 의해서나 가능하며 지원 데이터베이스 시스템의 형태와 질의 언어가 관계형 데이터베이스 관리 시스템 및 SQL로 제한되어 있기 때문에 객체지향 데이터베이스 시스템에 존재할 수 있는 기업의 제품 데이터나 설계도면 데이터의 직접적인 정보 공유가 불가능하다.

GDMS는 전역 스키마의 생성을 위해 스키마 통합을 해주어야 하며, 스키마 통합시에 발생하는 의미 충돌이나 이름 충돌을 해소하기 위한 수동적인 작업이 필요하다.

GDMS는 사용자에 의한 전역 질의와 질의 결과를 가져오기 위한 지역 질의들간의 통신이 소켓 기반으로 이루어지기 때문에 응용 시스템간의 저급의 데이터의 패킷 전송만을 허용하며 서버와 클라이언트측에서 메시지 교환을 위해 부호화(encode) 및 해석(decode)할 수 있는 응용 시스템 레벨 프로토콜을 작성해야 한다. 특정 프로토콜에 따라 사용자의 요청을 정확히 해석하기 위해 송수신되는 바이트들의 시험이 요구되어 프로토콜의 설계가 복잡하며 오류의 확률이 높다. 데이터 형식과 프로토콜이 특정 응용 시스템에만 적용되기 때문에 재사용이 어려우며 기업 통합이나 가상 기업에 있어서 확장성이나 유연성을 제공하지 못한다.

NCALS에서 구현한 수평적 분산 데이터베이스 시스템 형태의 통합 데이터 환경을 지원하는 시스템은 각 기업에 있는 지역 데이터베이스 시스템의 공유 정보에 대한 교환 색인 테이블을 중앙에서 관리하지 않기 때문에 각 지역에서 공유 정보에 대한 중복된 교환 인덱스 테이블들을 관리하고 있어야 한다. 기업간의 통신이 원거리 통신 네트워크(WAN) 기반으로써 네트워크가 항상 온라인 상태가 아닌 상황이기 때문에 색인 테이블들에 대한 동기적 갱신이 불가능하며 따라서 공유 정보에 대한 일관성(consistency)가 유지되기 어렵다.

NCALS에서 구현한 수평적 분산 데이터베이스 시스템의 사용자 인터페이스는 HTML(HyperText Markup Language) 문서의 폼(form)을 이용하는 CGI(Common Gateway Interface)를 사용하기 때문에 극히 제한적인 대화형 프로그램이 가능하며, 사용자의 질의 요청시 세션 연결이 불가능하여 매번 데이터베이스를 접속하기 때문에 서버에 대한 부하(overhead)가 크며 일관성 있는 세션 관리가 어렵다. 또한 외부의 시스템에 따라 각각의 프로그램을 작성해야 하는 단점이 있다. 관계형 데이터베이스 시스템을 기반으로 하며 CALS 표준 데이터를 수용하는 객체지향형 데이터베이스 시스템과의 연동을 고려하지 않고 있다.

NCALS의 수평적 분산 데이터베이스 시스템에서 제공하는 데이터의 형태는 단순한 파일로서 존재하며 이러한 파일들을 관리하기 위해 관계형 데이터베이스 시스템인 오라클을 사용한다. 즉, NCALS의 수평적 분산 데이터베이스 시스템에서는 동질 뿐만 아니라 이질의 데이터베이스 관리 시스템에서 제공하는 데이터들은 고려하지 않으며 단순히 파일들만을 공유되어지기 위한 데이터들로 고려한다.

2.3 통합 데이터 환경을 지원하기 위한 시스템의 고려 사항

가상 기업의 데이터 공유 지원 시스템를 분석하기 위해 기존의 CALS 통합 데이터 환경을 지원하는 시스템으로써 JCALS의 GDMS와 NCALS의 분산 데이터베이스 시스템의 문제점을 분석한 결과, 이질적인 분산 데이터들의 논리적 통합에 의한 통합 데이터 환경을 지원하는 시스템은 가상 기업의 플랫폼이 될 수 있는 인터넷 환경에서 사용자 질의 요청에 대해 전역적인 뷰(view)를 제공할 수 있도록 하며 일관성 있는 세션 관리가 되어야 한다. 각 기업의 자치성을 보장하며 기존에 사용중인 시스템을 유지하기 위해 참여한 데이터베이스들의 동적인 스키마 통합에만 의존하지 않으며 기업간의 계약이나 정책에 따라 자동적으로 관리될 수 있는 영역에서 부분적 스키마 통합을 허용하

여야 한다. 각 기업에 존재하는 이질의 하드웨어, 운영체제, 네트워크를 포함하는 플랫폼 계층에 대한 상호 운용성 및 확장성과 유연성을 제공할 수 있기 위하여 CORBA와 같은 표준을 수용하는 통신 미들웨어가 요구된다. 기업간에 존재하는 파일 형태의 데이터 뿐만 아니라 관계형 및 객체지향형 데이터베이스 시스템을 수용할 수 있어야 한다. 사용자와의 인터페이스를 위하여 다수의 사용자에 의해 동시에 데이터베이스를 처리할 수 있도록 설계되어 져야 하며 관계형 및 객체지향형 데이터베이스의 질의를 처리할 수 있는 객체형 질의 언어가 요구된다.

3. 데이터베이스 브로커 시스템

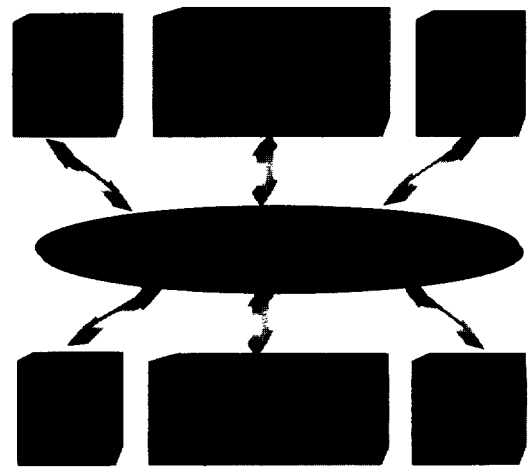
본 장에서는 통합 데이터 환경을 지원하는 시스템으로 전역 스키마를 생성하지 않고 통합된 전역적인 뷰를 제공하며 투명하게 데이터를 접근할 수 있는 데이터베이스 브로커 시스템을 제안한다.

데이터베이스 브로커 시스템은 데이터베이스의 전위기로써 데이터베이스 관리기를 호출하는 응용 시스템 소프트웨어와 데이터베이스 관리기 사이에서 인터페이스 계층의 역할을 담당한다. 또한 응용 프로그램 소프트웨어와 최종 사용자들에게 저장된 데이터의 위치 투명성 및 논리적 뷰를 제공하며 저장된 데이터는 데이터베이스 관리 시스템이나 일반 파일들에 대한 공통적인 인터페이스를 제공하고 공통된 질의 문장을 각 기업에 존재하는 지역 질의어로 해석할 수 있는 기능을 제공한다.

제안된 데이터베이스 브로커 시스템에서 제공하는 기능으로는 사용자의 계정 및 접근 제어를 담당하는 사용자 관리, 사용자의 질의 요청에 의해 생성된 공통 질의를 각 기업의 지역 질의 서버가 해석할 수 있도록 부질의(sub-query)로 분해하며 전달하는 전역 질의 서버, 분해된 부질의를 전달하기 위한 데이터 원시 정보 인터페이스 관리, 부질의를 각 지역 질의에 적합하게 해석하여 처리하는 지역 질의 서버 및 참여한 데이터 원시 정보(source) 들을 관리하기 위한 데이터 원시 정보 관리 기능들로 구성된다.

제안된 데이터베이스 브로커 시스템은 인터넷 환경에서 Java를 사용하여 사용자가 데이터베이스 접속에 의해 질의 요청시 일관성 있는 세션 관리가 유지되게 하며, 기업간의 이질 플랫폼상의 상호 운용성 및 위치 투명성을 제공하기 위해 통신 미들웨어로써 CORBA (Common Object Request Broker Architecture)를 사용한다. 클라이언트에 제공되는 공통 질의 언어는 객체지향 데이터베이스 및 관계형 데이터베이스를 수용할 수 있는 ODMG(Object Data Management Group) 표준의 객체지향 질의어(OQL : Object Query Language)를 사용한다. 이러한 데이터베이스 브로커 시스템은 사용자에 의해 요구된 데이터와 데이터 모델 및 플랫폼에 있어서 투명성이 제공되어 기업간의 정보 공유가 효율적으로 이루어지도록 하며 지역 데이터들의 자치성 보장 및 정보 검색에 있어서 고도의 투명성을 제공한다.

본 논문에서 제안한 데이터베이스 브로커 시스템의 구조는 다음과 같다(그림 3).



(그림 3) 데이터베이스 브로커 시스템 구조
(Fig. 3) Architecture of Database Broker System

3.1 데이터베이스 브로커 시스템의 구조

(1) 통합 데이터 엑세스 인터페이스

다수의 사용자와 등급에 따른 엑세스 제어 관리 기능을 제공한다. 기본적으로 사용자 ID와 패스워드, 사용자가 소속된 기관이 제공하는 정보의 등급과 제공받을 수 있는 정보의 등급에 따라 엑세스 제어의 등급이 결정된다.

사용자의 질의를 요청하기 위해 세션이 개설될 때 고유의 세션 ID를 가진 세션 서버가 생성된다. CORBA에서 제공하는 다중쓰레드를 사용하여 동시에 다수의 사용자에 의한 세션 처리가 가능하도록 한다.

(2) 데이터 원시 정보 인터페이스 관리기

사용자 질의에 따라 지역의 데이터 원시 정보들과 연결하기 위한 객체의 생성 및 관리를 담당하며 질의를 수행하는 전역 질의 서버와 지역 질의 서버와의 인터페이스 역할을 담당한다.

데이터 원시 정보 인터페이스 관리기는 접속 관리 기능과 접속 서버 기능으로 분리되어 세션 서버로부터 요청된 질의를 지역 질의 서버로 전달하여 질의 결과를 전달하는 역할을 담당한다. 즉, 세션 서버로부터 사상(mapping) 정보의 요청 및 파일 정보에 대한 요청을 받아 전역 질의 서버에 의해 분해된 부질의를 각 지역 질의 서버에 전달한다.

(그림 4)는 전역 질의를 부질의로 분해하여 데이터 원시 정보 인터페이스 관리기에 의해 지역 질의 서버에 전달하기 위한 인터페이스 및 각 지역 데이터베이스 시스템과의 사상을 정의한 ODL의 예제이다.

```
Interface DB extends DBtable {
  extent table_id
  key table_name
  attribute string table_type {RDBMS, FILE,
            OODBMS) ;
  attribute integer table_index_no
  atribute string table_create_date
  relationship
}
 mapping DB {
      origin DB1 (Oracle) : a_DB a;
      origin DB2 (Sybase) : b_DB b;
      origin DB3 (ObjectStore) : c_DB c;
      def_ext_DB_ext as
           select * from a, b, c
      def attr a. table_type as
           select a.table_type from a;
      def_attr b.table_index_no as
           select b. table_index_no from b;
      def_attr c.table_create_date as
           select c.table_create_date from c;
};
```

(그림 4) 전역 질의/지역 부질의 인터페이스를 위한 ODL
(Fig. 4) ODL for Global Query/Local Subquery Interface

(3) 데이터 원시 정보 색인 관리기

데이터 사전/디렉토리의 역할을 담당하는 데이터 원시 정보 색인 관리기는 클라이언트로부터 입력된 전역 질의가 전역 질의 서버에 의해 부질의로 분해된 후 각 부질의 정보를 가지고 있는 지역 질의 서버로 전달되기 위해 저장된 데이터베이스 및 파일 정보 등에 대한 사상 정보와 각 파일 정보를 저장 및 관리하는 역할을 담당한다. 또한 각 기업에서 참여하는 사용자의 정보를 관리한다. 사용자 정보, 사상 정보 및 파일 정보는 각 지역 데이터베이스 관리자나 권한이 부여된 사용자에 의해 등록되어 관리된다. 사용자 정보를 구성하는 객체는 사용자 ID, 사용자 패스워드, 사용자가 접근할 수 있는 제어 레벨 등에 대해 관리한다. 사상 정보를 구성하는 객체는 전역적으로 사용되는 메타 데이터명, 질의 요청때 제공되는 서비스명, 사상 인식자, 속성명, 지역 서버 IP 주소, 및 지역 데이터베이스명 등으로 이루어 진다. 파일정보를 나타내기 위한 객체는 파일명, 파일 주소, 파일 타입, 파일 설명, 파일의 내용에 대한 버전 및 파일 등록일, 기관명 등으로 구성된다. 이외에도 각 지역 질의 서버로부터 서비스 받기 위한 정보들을 등록하는 역할을 담당한다. 전역 질의 서버에 의해 요청되는 정보를 제공하여 지역 질의 서버의 질의를 전달하여 결과를 얻기 위한 IDL은 다음과 같다.

```
module DataSourceIndexManager {

  exception IndexManagerException {
       string reason; long type; };
  typedef sequence<string> seq_String;
  typedef sequence<seq_String> seq_seq_String;
  struct Location {
   string   str; //name of the LQSServer,     };
  struct Mapping {
   string   gtab_nm;   // global table name
   string   gfld_nm;   // global field name
   Location ldb_nm;    // local DB name
   string   ltab_nm;   // local table name
   string   lfld_nm;   // local field name
  };
  typedef sequence<Mapping>     seq_Mapping;
  struct JoinField {
   string              gtab_nm; // global table name
   seq_String  gfns;            // join fields
  };
  typedef sequence<JoinField>     seq_JoinField;
  struct BObject {              // Broker Object
   string   name;
   string   type;
   string   description;
   string   location;
  };
```

```
typedef sequence<BObject> BObjectSeq;
struct BOField {
 string   name;
 string   type;
};
typedef sequence<BOField> BOFieldSeq;
struct User {
 string   name;
 string   password;
 string   description;
 };
typedef sequence<User> UserSeq;

interface DataSourceIndexManager {
seq_seq_String        executeQuery(in string query,
            in seq_Mapping mappings,
            in seq_JoinField join_fieldnames)
      raises (DataSourceIndexManagerException);
seq_String  getMetaData()
      raises (DataSourceIndexManagerException);
void     open()
      raises (DataSourceIndexManagerException);
void     close()
     raises (DataSourceIndexManagerException);
BObject getBObject(in string name)
     raises (DataSourceIndexManagerException);
void    setBObject(in BObject anentity)
     raises (DataSourceIndexManagerException);
BObjectSeq getAllBObject()
     raises (DataSourceIndexManagerException);
BObjectSeq getBObjectByKeyword(in
     string keyword)
     raises (DataSourceIndexManagerException);
BOFieldSeq getBOField(in string e_name)
     raises      (DataSourceIndexManagerException);
void setBOField(in string e_name,in
       BOFieldSeq bofields)
     raises (DataSourceIndexManagerException);
User    getUser(in string name)
     raises (DataSourceIndexManagerException);
void        setUser(in User anuser)
     raises (DataSourceIndexManagerException);
UserSeq getAllUser()
     raises (DataSourceIndexManagerException);
 );

};
```

(그림 5) 데이터 원시 정보 색인 IDL
(Fig. 5) IDL for Data Source Index

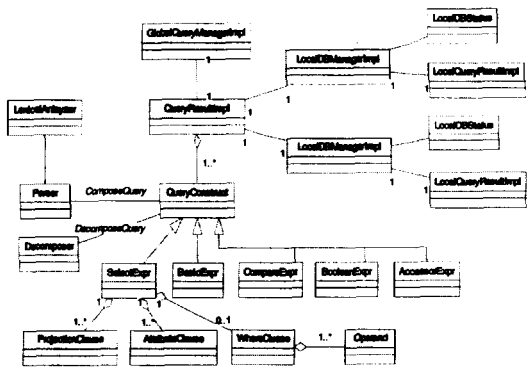
(4) 전역 질의 서버

전역 질의 서버는 세션 관리기에서 생성된 사용자의 질의를 OQL형식으로 입력받아 전역 질의 파싱 기술에 의하여 파싱된 결과를 부질의로 분해한 뒤, 해당 지역 데이터베이스 질의 관리기가 있는 서버로 전달하여 지역 질의 서버에서 얻어진 결과를 돌려 받아 화면에 출력시키는 과정을 책임진다.

전역 질의 처리기의 구성은 크게 어휘 분석기, 구문 해석기, 질의 생성기, 질의 분해기 및 질의 관리기능으로 구분된다.

사용자 인터페이스를 통해 입력받은 질의어는 OQL 형식을 따른다. OQL은 관계형 데이터베이스의 SQL에 대응하는 문법으로 구성된다

본 논문에서는 사용자 인터페이스로부터 전역 질의문을 OQL형식으로 입력 받아 지역 데이터베이스 질의 서버로 전달하기 위해 Java용 LALR 파서인 Java-CUP(Constructor for Useful Parser) 및 Java 용 어휘 분석기 생성기인 JLEX 프로그램을 응용하여 간단한 OQL 파서를 제작하였으며, 구현된 기본 객체 모델은 다음과 같다.



(그림 6) 구현된 OQL 파서의 기본 객체 모델
(Fig. 6) Object Model for Implemented OQL Parser

전역 질의 서버에 접속하여 서비스를 받기 위해 필요한 IDL 파일은 다음과 같다(그림 7).

```
// Global Query Server.IDL
module GQS
{
exception GQSException
{
```

```
 string      reason;
};

typedef sequence<string> seq_String;
struct ServiceMapping {
 string      sname;                  // service name
 string      mname;                  // mapping name
};
 struct Location {
 string      location;   // location information, string
                    type may be ok!
};
typedef sequence<Location> seq_Location;

//
/Data Structures for processing query results
//
enum ResultType {STRING, BOOLEAN,
     BYTE, SHORT, INTEGER, LONG,
     FLOAT, DOUBLE, BYTES, DATA,
     TIME, TIMESTAMP, SCIISTREAM,
     UNICODESTREAM,BYNARYSTREAM,
     DATE, URL, NUMERATION, OBJECT,
     USER_DEFINED, UNKNOWN };
struct QRMetaData {     // QueryResultMetaData
short       type;  //Broker가 이해할 수 있는 타입
            //(type이 -1이면 user defined type)
string      user_type;  //user가 정의한 data 타입
short       user_length;//user-defineddata타입길이
string      field_name;
};
typedef    sequence<QRMetaData>
 seq_ QRMetaData;
struct QRData {        // QueryResultData
  string    field_name;
  any       value;
};
  typedef sequence<QRData> seq_QRData;
  typedef sequence<octet>          seq_Octet;

interface QueryResult {
 seq_ QRData        getData();
      QRData        getDataValueByName(in string
        field_name);
      QRData        getDataValueByIndex(in
                 short index);

 string             getString(in short index);
 boolean            getBoolean(in short index);
 octet              getByte(in short index);
 short              getShort(in short index);
 long               getLong(in short index);
 float              getFloat(in short index);
 seq_Octet getBytes(in short index);
};


interface ResultSet {
 seq_QRMetaData     getDataSchema();
 QRMetaData         getDataschemaValue(in string
                      field_name);

 boolean    IsNext();
 QueryResult        getNext();

};
typedef sequence<ServiceMapping>
        seq_ServiceMapping;

interface GlobalQueryManager {
          ResultSet  executeQuery(
         in string oql_query,
         in seq_String services,
         in seq_ServiceMapping smappings,
         in seq_Location locations)
           raises (GQSException);
          };

interface GQSServer {
          GlobalQueryManager   open();
void      close(in GlobalQueryManager gqm);

 };
};
```

(그림 7) 전역 질의 서버를 위한 IDL
(Fig. 7) Global Query Server IDL

(5) 지역 질의 서버

지역 질의 서버는 전역 질의 서버로부터 전달받은 전역 질의의 부질의문인 오라클이나 사이베이스와 같은 각 지역의 관계형 DBMS의 데이터 ObjectStore 등과 같은 객체지향형 DBMS 및 파일 정보를 처리하기 하기 위해 질의를 변환하여 처리하고 질의된 결과를 전역 질의 처리기에 전달하는 역할을 담당한다.

본 논문에서 구현된 시스템에서는 각 지역의 관계형 DBMS인 오라클을 위해 JDBC Thin Driver를 사용하였으며 사이베이스를 위해서는 JDBC FastForward Driver를 사용하여 질의 처리를 한 후 질의 결과를 전

역 질의 서버에 전달하며 각 지역의 관계형 데이터베이스의 자료를 가져오기 위해 JDBC를 이용하여 질의 결과를 가져오는 처리과정의 IDL은 다음과 같다.

```
module LQS            // Local Query Server
{
 exception LQSException
 {
  string    reason;
 };
 typedef sequence<string> seq_String;
typedef sequence<octet>  seq_Octet;
//
//Data Structures for processing query results
 enum RT { BrokerT_STRING,
        BrokerT_BOOLEAN,BrokerT_SHORT,
        BrokerT_INTEGER,BrokerT_LONG,
        BrokerT_FLOAT,BrokerT_DOUBLE,
        BrokerT_BINARY,BrokerT_DATE,
        BrokerT_TIME,BrokerT_TIMESTAMP,
        BrokerT_OBJECT,BrokerT_
        USERDEFINED }; // QueryResult Type
//-------------------------------------
// idl type  Broker type       java type
//-------------------------------------
typedef     string      Broker_STRING;
typedef     boolean     Broker_BOOLEAN;
typedef     short       Broker_SHORT;
typedef     int         Broker_INTEGER;
typedef     long        Broker_LONG;
typedef     float       Broker_FLOAT;
typedef     double      Broker_DOUBLE;
typedef     seq_Octet   Broker_BINARY;

struct Broker_DATE { // jsava,sql.Date
            long year;
            long month;
            long day;
};
struct Broker_TIME { // java.sql.Time
            long hour;
            long minute;
            long second;
};
struct Broker_TIMESTAMP{
            long year,month,day;
            long hour,minute,second;
            long nano;
            };
typedef    seq_Octet    Broker_OBJECT;
typedef seq_Octet Broker_USERDEFINED;
struct RRField{          // Field of Query ResultRecord
            long         attr;
            any          value;
};
typedef sequence<RRField> RRecord;
struct RMDField {
string      field_name; // 필드 명
string      table_name;// 필드의 테이블 이름
RT          type;       // Broker가 이해할수있는 타입
string      user_type;    // user정의 data 타입
long        user_length;  // 해당data 타입  길이
};
typedef sequence<RMDField> RMetaData;

interface QueryResult {
RMetaData      getMetaData()
                  raises (LQSException);
RMDField  getRMDField(in long index)
                  raises (LQSException);
RMDField  getRMDFieldByName(in string field_name)
raises (LQSException);
RRecord    getRecord()
             raises (LQSException);
RRField    getRRField(in long index)
             raises (LQSException);
RRField    getRRFieldByName(in string field_name)
             raises (LQSException);
long       getColumnCount()
             raises (LQSException);
long       getRowCount()
             raises (LQSException);
boolean    IsNext()
             raises (LQSException);
boolean    next()
             raises (LQSException);
void       initCursor()
             raises (LQSException);
};
interface LocalQueryManager {
   QueryResult executeQuery(in string oql_query)
             raises (LQSException);
boolean    commit()
             raises (LQSException);
boolean    rollback()
             raises (LQSException);
};
interface LQSServer {
```

```
void        initLQSServer()
              raises (LQSException);
LocalQueryManager open(in long long ts)
                  raises (LQSException);
void        close(in LocalQueryManager lqm)
              raises (LQSException);
 };
};
```

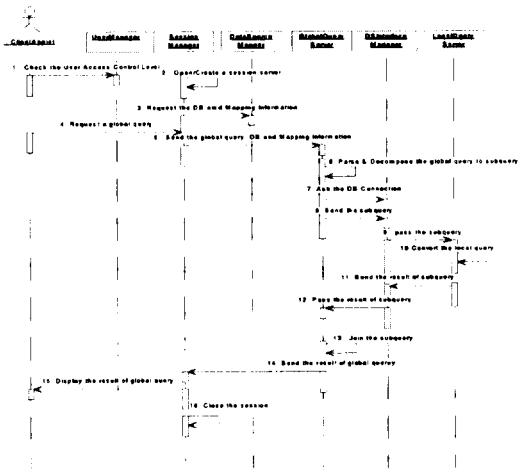
(그림 8) 지역 질의 서버를 위한 IDL
(Fig. 8) IDL for Local Query Server

객체지향형 데이터베이스 관리 시스템인 ObjectStore의 경우, 질의 처리를 위하여 특정의 드라이버를 사용하지 않고 ObjectStore에서 제공하는 Java 언어 클래스 라이브러리를 사용하여 바인딩함으로써 객체 질의 언어로 입력 받은 질의를 위한 질의 처리를 수행한다.

3.2 데이터베이스 브로커 시나리오

제안된 데이터베이스 브로커의 수행절차는 다음과 같다(그림 9).

1) 사용자 인터페이스에 의해 사용자의 등급에 따른 엑세스 제어 절차에 따라 질의 입력이 허용된다.
2) 세션 관리기에 의해 세션 서버가 생성된다.
3) 세션 서버는 데이터 원시 정보 관리기에 지역 질의 서버에 사상하기 위한 정보 및 파일 정보를 요청한다.
4) 세션 서버는 사용자에게 전역 질의를 요청 받는다.
5) 세션 서버는 요청 받은 전역 질의를 전역 질의 서버에 전달한다.
6) 전역 질의 서버는 전역 질의를 구문 해석하여 부질의를 생성한다.
7) 전역 질의 서버는 데이터베이스 인터페이스 관리기에 지역 질의 서버와의 접속을 요청한다.
8) 전역 질의 서버는 데이터 원시정보 인터페이스 관리기에 부질의를 전달한다.
9) 데이터 원시 정보 인터페이스 관리기는 해당 지역 질의 서버에 부질의를 전달한다.
10) 지역 질의 서버는 전달 받은 부질의를 지역 질의 처리를 위해 변환하여 부질의를 처리한다.
11) 지역 질의 서버는 부질의 결과를 데이터 원시 정보 인터페이스 관리기에 전달한다.
12) 데이터 원시 정보 인터페이스 관리기는 부질의 결과를 전역 질의 서버에 전달한다.
13) 전역 질의 서버는 부질의 결과들을 조인 처리한다.
14) 전역 질의 서버는 세션 관리기에 전역 질의 결과를 전달한다.
15) 세션 관리기는 전역 질의를 사용자 애플릿에 나타낸다.
16) 세션 관리기는 데이터베이스 연결을 해제하고 세션을 종료한다.



(그림 9) 데이터베이스 브로커 처리 절차
(Fig. 9) Database Broker Process Steps

3.3 구현 환경

전역 질의 서버와 오라클, 사이베이스 및 ObjectStore를 위한 지역 질의 서버는 PC상의 Widnows 95 환경에서 실행되며 데이터 원시 정보 인터페이스 및 색인 관리기는 Windows NT에서 실행된다. 또한 오라클, 사이베이스 및 ObjectStore의 데이터베이스 관리 시스템은 Solaris 2.5.1 운영 체제를 사용하는 워크스테이션에 설치되어 있다. 이들을 사용하는 시스템은 사용자 인터페이스로 Java 애플릿이 Windows 95 환경에서 전역 질의 서버 인터페이스를 통해 CORBA와 접속하게 된다. 또한 오라클과 사이베이스, ObjectStore로 구성된 각 지역 질의 서버는 Windows 95 환경으로 데이터베이스 인터페이스를 통해 부질의를 전달 받고 마찬가지로 CORBA를 사용하여 데이터베이스 인터페이스를 통해 결과를 전달한다.

데이터베이스 브로커 시스템의 구현 및 사용 환경은 다음과 같다<표 1>.

〈표 1〉 데이터베이스 브로커 개발/사용 환경
〈Table 1〉 Development/Use Environment for Database Broker

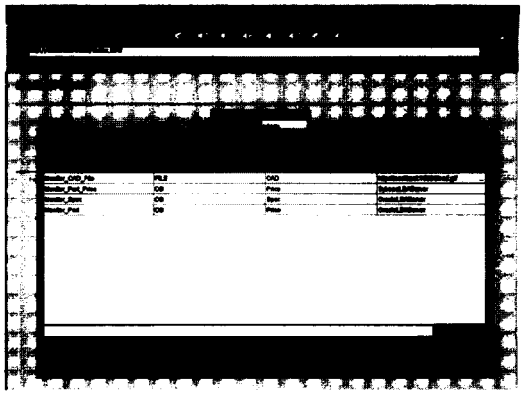
구 분	항 목	비 고
소프트웨어	설계 도구	Rational Rose for Java
	개발 언어	Java JDK 1.1.5, Java JFC Swing 1.02
	CORBA	Visibroker 3.1, Gatekeeper, OSagent
	JDBC 드라이버	Oracle용 : JDBC Thin Driver Sybase용 : FastForward 3.1.1
	웹 서버	Apache서버
	웹브라우저	Netscape 4.05 이상, MS Explor 4.0 이상, Hotjava
	데이터베이스 시스템	Oracle 7.3.4, Sybase 1.1.0, ObjectStore
하드웨어	PC	펜티엄 PC, Windows NT Server
	워크스테이션	SUN Ultra Sparc
운영체제	PC	Windows95, Windows NT 4.0
	워크스테이션	UNIX

3.4 사용자 인터페이스

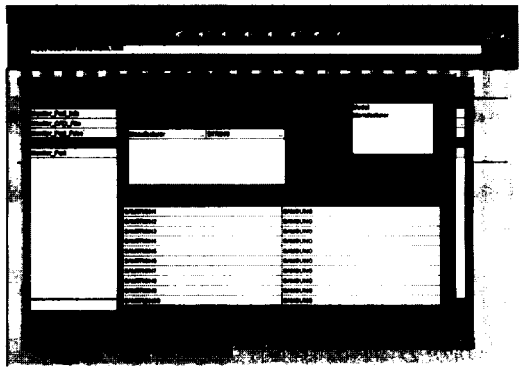
구현된 데이터베이스 브로커 시스템의 사용자 인터페이스는 사용자 로그인 인터페이스, 사용자 정보, 매핑 정보 및 파일 정보 등을 등록 및 관리하기 위한 데이터 원시 정보 색인 관리기 인터페이스 및 전역 질의를 입력하여 결과를 디스플레이하기 위한 세션 관리기 인터페이스로 구분한다.

(그림 10)은 세션 관리기 인터페이스로 제공되는 객체테이블의 이름, 타입, 객체 테이블의 간단한 설명과 위치정보를 나타내는 화면이며 키워드 선택을 사용하여 객체 테이블의 설명 필드 중 키워드와 일치하는 객체 테이블의 검색도 가능하다.

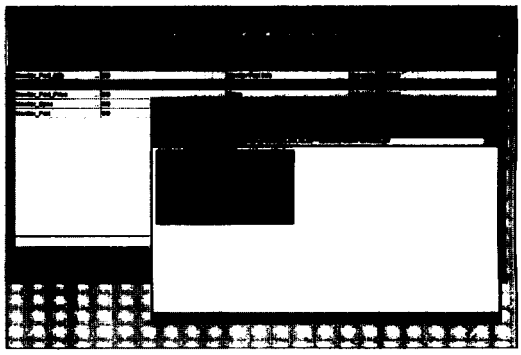
객체 테이블이 선택되면 해당 객체 테이블에서 제공되는 필드이름과 필드 타입을 제공하는 패널과 선택된 필드를 나타내는 패널 및 선택된 필드들에 대한 질의 전송의 결과를 보여주는 패널로 구성되며(그림 11), 선택된 객체가 파일 타입일 경우 URL이나 File 위치 정보에 따라 파일 타입에 해당되는 응용소프트웨어를 통해 디스플레이된다(그림 12).



(그림 10) 객체 테이블 선택 화면
(Fig. 10) Object Table Selection Screen



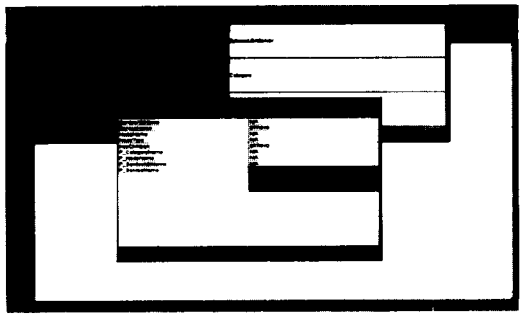
(그림 11) 선택된 객체 테이블에 대한 정보 화면
(Fig. 11) Selected Field Data Display Screen



(그림 12) 파일 타입 정보 화면
(Fig. 12) File Type Data Screen

(그림 13)은 데이터 원시 정보 색인 관리기 인터페이스로써 각 지역 데이터베이스의 매핑 정보, 사용자

정보 등을 등록하는 인터페이스로써 객체 테이블의 필드들을 선택하여 입력 시켜주는 화면이다.



(그림 13) 데이터 원시 정보 색인 관리기 인터페이스
(Fig. 13) Data Source Index Manager Interface

3.5 기대 효과

본 논문에서 제안되어 구현된 데이터베이스 브로커 시스템의 기대 효과를 분석하기 위해 통합데이터 환경을 지원하는 시스템으로 JCALS의 GDMS와 NCALS의 수평적 분산 데이터베이스 시스템을 비교 분석하였다<표 2>.

〈표 2〉 JCALS GDMS, NCALS 수평적 분산 데이터베이스 시스템 및 제안 시스템의 비교
〈Table 2〉 The comparison of JCALS, NCALS and Proposed System

구 분	JCALS GDMS	NCALS 수평적 분산 DB 시스템	제안 시스템
분산 DB 형태	수직적 분산 (Top-Down)	수평적 분산 (Bottom-Up)	혼합 분산 (Hybrid)
소스 데이터 형태	RDB, 화일	RDB, 화일	RDB, OODB, 화일
사용자 인터페이스	윈도우 기반 GUI	CGI	JAVA
사용자 액세스 제어	사용자ID, 암호	사용자ID, 암호	사용자ID, 암호
지원 질의 언어	SQL	SQL	SQL
글로벌 스키마	통합	통합하지 않음	통합하지 않음
질의 분해기	지원	지원하지 않음	지원
질의 통합기	지원	지원하지 않음	지원
질의간의 통신	소켓통신	CORBA	CORBA

본 논문에서 제안된 데이터베이스 브로커 시스템의 분산 데이터베이스 형태는 JCALS의 GDMS가 중앙 집중적 통제에 의한 수직적 분산 형태로써 새로운 시스템을 구축한다는 단점과 NCALS의 시스템은 수평적 분산 데이터베이스를 유지하기 위해 각 지역에서 공유 정보에 대한 교환 색인 정보를 중복되게 관리하는 단점을 극복하기 위해 공유 정보들에 대한 데이터 원시 정보를 전역적으로 한 저장소에 등록 및 관리하여 스키마 통합의 필요성 및 중복 관리에서 파생되는 데이터의 일치성을 보장하기 어려운 문제점을 제거하고 각 기업에서 사용중인 시스템의 자치성을 보장하는 형태인 혼합 분산 데이터베이스 형태를 기반으로 하여 CALS의 기본 개념인 기존 시스템을 유지할 수 있도록 하였다.

제공되는 원시 데이터의 형태가 JCALS의 GDMS는 관계형 데이터베이스와 파일 형태의 데이터만을 고려하며 객체 지향형 데이터베이스 시스템에서 관리되는 도면 데이터나 객체 지향형 데이터에 대한 고려를 하지 않고 있으며 NCALS의 수평적 데이터베이스는 파일 형태의 데이터만을 취급하며 단지 이러한 파일 데이터들을 관리하기 위해 관계형 데이터베이스 시스템인 오라클을 사용하는 정도로써 고급의 데이터베이스 관리 시스템의 데이터 원시 정보에 대한 직접적 접근을 고려하지 않는 것과 비교하여 제안 시스템은 관계형 데이터베이스, 객체지향형 데이터베이스 및 파일 형태를 지원하는 데이터 모델 및 질의 언어를 수용하여 관계형 데이터베이스, 객체지향형 데이터베이스 및 파일 형태의 데이터 원시 정보의 직접적 접근이 가능하도록 구현되었다. 또한 이러한 데이터 원시 정보들을 질의하기 위한 질의 언어로써 JCALS의 GDMS와 NCALS의 수평적 분산 데이터베이스 시스템을 관계형 질의 언어인 SQL을 지원하나 본 논문에서 제안된 데이터베이스 브로커 시스템을 관계형 및 객체지향형 데이터베이스 시스템 뿐만 아니라 파일 시스템의 질의가 가능한 객체지향 질의 언어(OQL)을 지원하도록 구현되었다.

클라이언트를 위한 사용자 인터페이스로써 JCALS의 GDMS는 윈도우 기반의 그래픽 사용자 인터페이스를 제공하는 반면 NCALS의 수평적 분산 데이터베이스는 CALS의 사용자가 전세계적일 수 있다는 점을 고려하여 인터넷 환경을 고려하였으나 CGI로 구현되어 CGI의 단점인 일관성 있는 세션 관리가 유지 될 수 없어 이에 따른 부하를 감수해야 한다. 이러한 단점을 극복하기 위해 본 논문에서 제안한 데이터베이스 브로커 시스템은 인터넷 환경에서 상호 대화형 응용

프로그램을 지원하며 일관성 있는 세션 관리가 가능한 Java를 사용하여 구현하였다.

JCALS의 GDMS가 하향식 분산 데이터베이스 형태를 취하여 전역 스키마를 통합하는 방식으로 지역의 시스템에 대한 자치성에 제한이 되는 반면 본 논문에서 제안한 데이터베이스 브로커 시스템은 전역 스키마의 통합을 실행하지 않고 지역의 기존 시스템의 자치성을 유지 시켜주는 장점을 가진다.

JCALS의 GDMS는 전역 질의 서버와 지역 질의 서버간의 통신을 소켓 기반으로 하기 때문에 각 지역 질의 서버간의 통신 프로토콜을 개별적으로 설계 및 구현해야 하기 때문에 복잡성을 띄며 지역 질의 서버의 확장성이나 유연성을 제공하지 못하나 본 논문에서 제안한 데이터베이스 브로커 시스템은 표준을 수용하는 CORBA를 사용하기 때문에 글로벌 질의 서버와 지역 질의 서버간의 상호 운용성과 위치 투명성은 물론 유연성 및 확장성을 제공한다.

위의 분석 결과를 토대로 본 논문에서 제안한 데이터베이스 브로커 시스템은 CALS의 기본 개념인 기존 시스템을 사용하면서 각 기업간에 존재하는 지역 데이터베이스 응용 시스템의 자치성을 유지하며 통합 데이터 환경을 지원하기 위해 본 논문의 제2장에서 분석한 통합 데이터 환경을 지원하기 위한 시스템의 고려 사항을 충족하며 기존 시스템에서 제공하지 못한 기술적 문제점들을 해결하여 구현되었다.

4. 결 론

본 논문에서는 "가상 기업의 실현"이라는 CALS의 궁극적인 목적에 따라 기업간의 정보 공유를 위하여 기존에 사용 중인 시스템을 유지하면서 위치 투명성이 제공될 수 있는 통합 데이터 환경을 지원하는 데이터베이스 브로커 시스템의 프로토타입을 구현하였다.

본 논문에서 구현된 가상 기업의 데이터 공유 지원을 위한 핵심 기술인 통합 데이터 환경을 지원하는 데이터베이스 브로커 시스템은 기존의 시스템을 그대로 유지하여 기업의 자치성을 인정하며 이질 플랫폼상에서 물리적으로 분산된 데이터 원시 정보들을 논리적으로 통합하여 사용자에게 위치 투명성을 제공하며 단일의 뷰로써 정보가 제공되어 지도록 하여 "한번 생성된 데이터는 다수의 기업에 의해 공유되어 사용되어 져야 한다"는 가상 기업의 목적을 만족하는 시스템으로 제공될 수 있다.

현재의 프로토타입 시스템에 기업간의 민감한 데이터의 공유를 위해서 필요한 고급의 보안 및 인증 기능 등이 추가되어야 할 사항이다.

참 고 문 헌

[1] Department of Defence : JCALS Infrastructure Capabilities Description Release 3.1, Computer Sciences Corporation, January 1998.

[2] CSC, JCALS Developers Tool Kit, GDMS Implementation Guide, Second Release, January 1997.

[3] A. Dogac, C.Dengi and T.Ozsu, "Building Interoperable Database Nabagenebt Systems on Distributed Object Management Platforms," Technical Report 96.2.1 Software R&D Center, METU, Jan. 1996.

[4] OMG, The Common Object Request Broker Architecture and Specification, Revision 2.0, July, 1996, http://www.omg.org

[5] Standard for Object Databases, IDC ODMG White paper, http://www.odmg.org/download/downloadpage.htm

[6] R.G.G. Cattell, Douglas K.Barry, The Object Database Standard : ODMG 2.0, Morgan Kaufmann Publishers, Inc., 1997.

[7] Akira Nishiguchi, Koichiro Shida, Yuji Takada, Steel Plant CALS Project, CALS Expo International 1997 Proceding, November, 1997. http://www.tcals.or.jp/paper/

[8] Y. Yamamoto, M. Sekiya, N. Mori, H. Yamaza, M. Kamita, Y. Teshima, "Implementing a Network Infrastructure for CALS Distributed Systems," CALS Expo International 1997 Proceeding, Nover, 1997.

[9] ISG Navigatorl, http://www.isg.co.il/navigatr.htm

[10] C. Mic Bowman, Peter B.Danzig, Darren R.Hardy, Michael R. Hardy, "The Harvest Information Discovery and Access System," http://harvest.transarc.com/

[11] E.Kilic, G.Ozhan, C.Dengi, N.Kesim, P.Koksal, A.Dogac, "Experience in Using CORBA for a

Multidatabase Implementation," Proc. of 6th intl. Workshop on Database and Expert System Applications, London, Sept. 1995.

[12] 이남용, 송운호, CALS/EC, 법영사, 1996.

[13] 우훈식, 정석찬, 윤선희, 조장혁, 주경준, "CORBA 환경하의 CALS 통합 데이터베이스 설계", pp.59-68, 한국 CALS/EC 학회 '97 종합학술대회, 1997.

[14] 윤선희, 주경준, 정진욱, "CALS 기반 분산 네트워크 하부구조에 관한 연구", 한국 정보과학회 '98 춘계학술대회, 1998.

[15] 윤선희, 우훈식, 문희철, 정승욱, 박상봉, "CORBA 기반 멀티데이터베이스 구현에 관한 연구", 한국정보처리학회 '98 춘계학술대회, 1998.

[16] 이강찬, 이규철, 진성일, "CALS 환경에서의 데이터베이스 통합에 관한 접근 방법의 연구", pp.233-242, 한국 CALS/EC 학회 '97 하계 학술대회, 1997.

[17] 안성진, 정진욱, "인터넷관리를 위한 네트 워크 설계지원시스템", pp.69-79, 정보통신기술, 한국정보과학회 정보통신연구회, 1997.

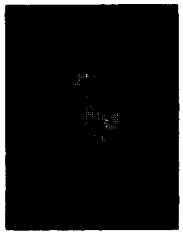
[18] 박재현, 코아 코바, 영한 출판사, 1998.

[19] 박치항, 이상구 역, 분산 오브젝트 지향 기술 CORBA, 홍릉과학 출판사, 1996.

[20] 김지운, 문강식, 김수웅, 이전영, "Web과 멀티데이터베이스 시스템의 CORBA 기반 연동", pp.323-328, HCI '97 학술대회 발표 논문집, 1997.

[21] 박현민, 이선미, 정효택, "객체지향 메타 모델의 관계형 데이터베이스 스키마 변환", pp.343-346, 한국정보처리학회, '97 추계 학술 발표논문집, 1997.

[22] 윤인중, 김홍남, 김창갑, "웹환경하에서 자바를 이용한 효율적인 데이터베이스 접속에 관한 연구", pp.451-456, 한국정보처리학회, '97추계학술대회, 1997.



윤 선 희

e-mail : shyoun@etri.re.kr

1983년 숭실대학교 전산학과(학사)

1986년 웨인주립대학교 전산학과(석사)

1997년 성균관대학교 정보공학과 박사과정 수료

1991년~현재 한국전자통신연구원, 컴퓨터S/W소프트웨어 기술연구소 선임연구원

관심분야 : 분산처리응용, CALS/EC, HCI, 네트워크 관리

정 진 욱

e-mail : jwchung@songgang.skku.ac.kr

1974년 성균관대학교 전기공학과(학사)

1979년 성균관대학교 전자공학과(석사)

1991년 서울대학교 계산통계학과(박사)

1973년~1985년 한국과학기술연구소 데이터통신연구실장

1985년~현재 성균관대학교 전기·전자 및 컴퓨터공학부 교수

관심분야 : 네트워크 관리, 네트워크 보안, 고속 및 무선 통신 프로토콜