□ 기술해설 □

분산객체 컴퓨팅과 ERP서버

한국전자통신연구원 박성진*·문봉교

1. 서 론

전사적 자원관리 또는 기업통합 정보시스템으로 불리는 ERP(Enterprise Resource Planning)는 국가 및 기업 정보인프라 구축을 위한 현실적인 대세이며 정보시스템의 총아로 인식되고 있다. 이것은 최근 정보기술의 눈부신 발전을 통해 기존 단위업무 및 부서별로 적용하던 정보시스템 구축이 전사적인 규모로 조직의 정보를 관리하는 수준에 이르게 되었기 때문이다. ERP시스템은 기업의 생산, 자재, 영업, 인사, 회계 등의 업무를 통합 관리해주는 대형 경영관리용 패키지 소프트웨어로서 사용자는 ERP시스템 패키지중에서 자신의 기업환경에 적합한 모듈만 부분적으로 선택하여 시스템을 유연하게 구축하거나 기존 시스템을 확장할 수 있다. 이러한 시스템 통합성, 유연성과 같은 ERP시스템의 장점으로 인해 기업은 정보시스템을 자체 개발해야 하는 부담을 줄일 수 있으며 ERP를 사내 정보인프라로 구축, 활용할 수 있다. 따라서, ERP는 발전된 정보기술을 바탕으로 모든 기업업무를 총괄하는 전사적 정보시스템으로서 그 입지를 굳혀가고 있다. 현재, 전세계 ERP시스템 시장의 34%를 점유하고 있는 독일의 SAP을 필두로 미국의 SSA(Systems Software Associates), ORACLE, AVALON, QAD 등 ERP 공급업체들이 속속 출현하고 있으며 가트너그룹은 2000년까지 전세계 기업의 40%가 ERP시스템을 도입, 적용하게 될 것이라고 예측하고 있다.

한편, 최근 ERP시스템에 있어 네트워킹 및 인터넷/웹 분야의 기술발전은 네트웍 중심의 새로운 컴퓨팅 모델을 제시하도록 요구하고 있으며, 이러한 컴퓨팅 환경하에서 많은 새로운 기술들이 대규모 어플리케이션의 분산 컴퓨팅 기반구조 선점을 위해 경쟁하고 있다. 여기에는 Thin Client, 저가의 개방형 플랫폼 등을 가능케하는 컴퓨팅 운영기술들이 포함되며 특히, 최근 들어 분산 컴퓨팅 기술을 활용한 3계층(3-tier)상의 ERP시스템 기술이 일반화되고 있다. 따라서, 분산 컴퓨팅 기반의 ERP시스템과 운영환경을 지원하는 ERP서버의 역할이 점차 강조되고 있으며 본 논문에서는 ERP서버의 현황과 분산객체 미들웨어를 활용한 ERP서버에 대해 기술하고자 한다.

2. ERP서버 현황

2.1 ERP시스템

ERP시스템(또는 ERP패키지)은 각 기업이 고유환경에 맞도록 자체 개발한 시스템과는 달리 기본적으로 모든 유형의 기업에 맞도록 개발된 표준 소프트웨어의 특성을 갖는다. 따라서, ERP시스템을 도입하기 위해서는 각 기업 현황에 맞도록 적합화(customizing)시키는 과정이 요구되며 이에 앞서 기존 업무처리 방식의 개선이라는 BPR(Business Processing Reengineering)이 효과적으로 선행되어야 한다.

선진외국 ERP시스템들은 분산객체 컴퓨팅 등 새로운 정보기술 패러다임을 기반으로 최신의 ERP시스템 기술을 개발하고 있다. 선진 ERP시스템들의 기술적인 특징은 다음과 같이

* 정회원

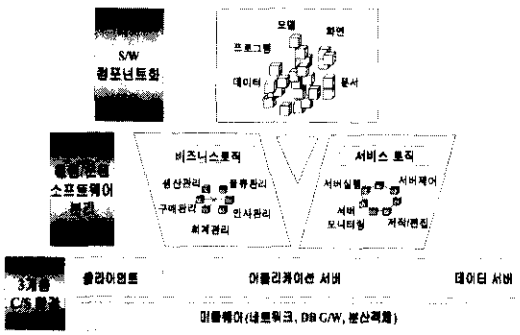


그림 1 선진 ERP 시스템의 특성

요약할 수 있다(그림 1 참조).

2.1.1 소프트웨어의 컴포넌트화를 통한 패키지형 시스템

다양한 업종별 또는 생산형태별 비즈니스 요구사항을 수용하기위해 철저히 컴포넌트(component) 단위로 소프트웨어를 개발하는 전략을 택함으로써 기업현장에 적용시 ERP시스템의 커다란 수정없이 적합화가 가능하도록 하고 있다. 이에 고품질, 저비용의 패키지 개발이 가능하여 시장 경쟁력이 강화되고 있다.

2.1.2 운영 및 응용 소프트웨어의 분리형 시스템

다양한 하드웨어 플랫폼과 운영체제 등에 효율적으로 적용하기위해 시스템운영 소프트웨어(개발도구 및 서버운영관리 소프트웨어)와 시스템응용 소프트웨어(비지니스 로직)를 분리하는 경향이 있다. 이는 이종의 운영환경에서도 시스템운영 소프트웨어의 일부 모듈만 수정을 하고, 방대한 응용 소프트웨어는 수정하지 않음으로써 기술동향과 시장동향 변화에 신속히 대응 가능하고, 시스템 이전 비용이 저렴한 장점이 있다.

2.1.3 3계층 클라이언트/서버(3-tier C/S)형 시스템

네트워킹기술의 급속한 발전과 활용기반 확대에 힘입어 확장성과 개방성이 우수한 3계층 C/S구조로 ERP시스템의 운영기반이 이동하고 있으며 모든 응용프로그램이 클라이언트에서 작동되는 기존 2계층 C/S구조에 비해 시스템 비용을 절감하고, 사용자 증가 정도에 따라 서버 시스템을 점진적으로 확장할 수 있다.

ERP시스템의 편리한 기능과 통합용이성으로 인해 국내 ERP시장 규모는 점차 커지고 있으나 독자적인 개발보다는 독일, 미국 등의 선진 ERP시스템의 도입 및 적합화에 중심이 실린 상황이다. 한편, ERP시스템을 도입하는 경우, 일부 기능이 국내 현실에 맞지않는 문제점이 있어 "한국형 ERP"라는 이름으로 여러 시스템들이 개발되고 있지만 아직까지 기존 MIS(Management Information System) 패키지 체제를 크게 벗어나지 못하고 있다.

특히, 고품질, 저비용의 소프트웨어 개발을 위한 소프트웨어 컴포넌트의 개념을 지원하고 고유한 소프트웨어 개발도구를 갖춘 선진 ERP시스템과는 달리 국내 ERP의 경우, PowerBuilder, Visual Basic, Delphi 등의 개발도구를 이용함으로써 소프트웨어 개발시 발생하는 결과물을 컴포넌트 형태로 통합관리하는데 한계가 있다. 또한, 자체 소프트웨어 개발도구의 미비로 인해 확장성과 개방성이 우수한 3계층 C/S 구조의 장점을 활용하지 못하는 문제점이 있다.

2.2 ERP 서버

메인프레임 중심의 중앙집중식 컴퓨팅환경과 제조, 생산, 회계, 인사, 자재관리 등을 독립적으로 개발운영하던 기업의 정보인프라는 최근 정보기술 발전에 힘입어 n계층 C/S환경과 개방형 ERP시스템으로 전환하고 있다. 이 과정에서 기존의 메인프레임 혹은 단일서버에만 존재하던 시스템 어플리케이션이 이제는 여러 서버에 분산되고 그에 따른 소프트웨어 유지보수 비용이 증가함에 따라 이기종간의 C/S시스템을 더욱 효율적으로 통합, 관리할 수 있는 ERP 서버의 중요성이 강조되고 있다.

C/S환경으로의 변화는 워크스테이션 혹은 PC 성능의 비약적인 향상, GUI 환경의 일반화, 관계형 DB의 발전에 따른 데이타와 프로그램 코드와의 분리 등에 기인한다. 이에 따른 이점으로는 개방형 시스템, 확장성과 가용성, 소프트웨어 생산성 및 성능의 향상을 들 수 있다. 가령, ERP C/S 시스템의 규모가 커지면 처리기능을 클라이언트가 전담하는데 한계가 있

음으로 클라이언트가 담당하는 처리 기능을 서버쪽에 분산시키는 형태가 자연스러우며 DBMS가 갖는 저장 프로시저(stored procedure) 기능을 이용해 서버쪽에서 일정부분 처리하는 것이 그 예이다. 또한, 현재는 대형 엔터프라이즈 어플리케이션이 늘어나고 다양한 시스템, 운영체제 등 이기종 플랫폼을 효율적으로 통합운영할 필요성이 커지면서 클라이언트와 DB 서버 사이에 미들웨어층을 두는 3계층 C/S구조로 ERP 운영환경이 자연스럽게 변환하고 있다. 3계층 C/S는 기존 2계층 구조에서 클라이언트에 포함돼 있던 응용 로직을 미들웨어층(혹은 어플리케이션 서버)으로 옮겨 전체 컴퓨팅 성능을 향상시켜줄 뿐만 아니라 다양한 이기종 플랫폼을 중간층에서 통합하고 클라이언트를 가볍게 만들어줌으로써 확장성과 유지보수성, 성능, 보안 및 관리 등에 이점이 있으며 대규모의 조직체에 적합한 구조로 알려져있다. 그러나, 2계층 구조에 비해 구축 비용이 많이들고 개발이 어렵다는 단점이 있다.

미들웨어(middleware)란 좁게는 클라이언트가 이질적인 서버들에 일관되게 접근하여 이들 서버의 서비스를 이용하게 해주는 기본통신 소프트웨어(네트워크 미들웨어)를, 넓게는 클라이언트와 서버사이의 상호작용을 위해 분산된 다양한 서비스를 지원하는 모든 소프트웨어(서비스 미들웨어)를 의미한다. 여기서 네트워크 미들웨어는 RPC(Remote Procedural Call)나 NOS(Network Operating System), Netware 등을 포함하며 서비스 미들웨어에는 마이크로소프트사의 ODBC, 볼랜드의 IDAPI, SAG(SQL Access Group)의 CLI(Call Level Interface), 오라클의 Glue와 같은 DB 게이트웨이 미들웨어와 Tuxedo, Encina와 같은 OLTP(OnLine Transaction Processing) 미들웨어, Orbix, ObjectBroker, PowerBroker, DCOM과 같은 분산객체 미들웨어가 속한다. 미들웨어는 클라이언트와 서버사이의 통신, 다양한 DB 인터페이스, 이기종 통신프로토콜의 연결, 보안 및 모니터링 기능 등을 수행하는 소프트웨어로 C/S환경이 급속도로 확산되면서 이기종 분산시스템 환경을 효율적으로 통합하기 위한 필수도구로 부상하고 있다. 이 가운데에서도 그동안 이기종 DB간의 트랜잭션 처리를 주로 담당하던 TP모니터계열의 제품이 미들웨어 시장의 주류를 이루어 왔으나 최근 들어 분산객체 미들웨어에 대한 관심이 높아지고 있다. 미들웨어를 통해 복수개의 클라이언트와 서버사이의 인터페이스를 해결함으로써 이질적인 대규모 분산 환경에서 별도의 인터페이스 프로그램 없이도 접근할 수가 있으므로 분산환경에서 미들웨어의 도입과 활용은 중요하다.

ORB(Object Request Broker)로 대변되는 분산객체 미들웨어는 상이한 각종 플랫폼의 객체기반 애플리케이션간의 통신을 중개하는 미들웨어로 분산시스템의 통합뿐만 아니라 개발된 코드의 재사용, 시스템 유지보수 및 향후 확장이 용이하다는 점 등 객체지향 기술의 이점을 지원하는 장점이 있다.

3. 분산객체 미들웨어

C/S환경에서 운영되는 프로그램은 본질적으로 분산된 환경에서 동작하는 프로그램이다. 컴포넌트 객체를 분산해 사용할 수 있도록 하는 분산객체 기술의 이점은 이질적인 플랫폼하에서의 상호운영성(interoperability)을 보장하며 네트워크를 통해 분산된 원격 시스템에 대해서도 객체 또는 컴포넌트 개념으로 접근할 수 있다는 것이다. 역으로 이야기하면 객체 또는 컴포넌트가 동일 머신 또는 이종의 다른 머신에 위치해도 정상적으로 동작할 수 있는 환경을 지원한다는 것이다. 이를 위해서는 소켓 API, RPC와 같은 네트워크 프로그래밍이 단순화되어야 하는데 이에 대한 해결책을 제시하는 것이 CORBA와 DCOM이다. 이들 방식은 동일한 주소 영역에 존재하는 것처럼 프로그래밍할 수 있는 컴포넌트 기반의 소프트웨어 구조를 지원해준다.

DCOM과 CORBA 모두 C/S방식의 통신을 지원하며 클라이언트는 서버의 서비스를 요청하기위해 메소드를 호출함으로서 원격 서버객체의 구현코드를 실행시킨다. 이때 요청한 서비스는 서버객체안에 구현되어있고 객체의 인터페이스는 IDL을 통해서 표현된다. 클라이언트는 IDL로 명시된 방법으로 서버의 메소드를

호출해 원하는 기능을 이용하게 되며 서버객체 구현에 관해서는 알지 않아도 된다. 여러개의 원격 컴포넌트로 구성된 분산응용 프로그램을 만들때 CORBA와 DCOM을 이용할 경우 많은 장점이 있다.

컴포넌트의 재사용으로 인한 비용 및 개발시간 감소, 위치 투명성, 개발 언어 및 플랫폼 독립성, 클라이언트와 서버의 효율적인 연결 관리, 기본적인 컴포넌트의 버전관리 및 보안 기능 등 많은 기능들을 분산객체 미들웨어를 통해 얻을 수 있다.

3.1 CORBA

CORBA는 800여개의 컨소시엄 구성업체(썬, IBM, 넷스케이프, 애플, 아이오나 등)가 제안한 분산객체 미들웨어로 CORBA의 핵심은 C/S환경하에서의 객체들의 대화통로인 객체버스 ORB(Object Request Broker)이다. CORBA는 ORB의 표준으로서 CORBA 1.1(1991년)은 IDL과 ORB를 통한 C/S 객체간의 상호작용 API를 정의하고 있다. CORBA 2.0(1994년)은 서로 다른 업체들간에 ORB가 어떻게 상호작용하는지를 정의함으로써 호환성을 보장하며 현재 버전 2.2까지 발표된 상태이다.

CORBA에서 특정 컴포넌트(클라이언트)가 다른 컴포넌트(구현객체)를 호출하게 되면 클라이언트는 IDL 스텁(stub)이나 동적 호출인터페이스를 통해서 메소드 호출을 ORB로 보낸다. 미리 컴파일된 코드인 스텁은 클라이언트가 어떻게 서버의 원하는 서비스를 호출하는가를 정의함으로써 객체 서비스에 정적인 인터페이스를 제공한다. 즉 클라이언트의 호출을 ORB가 알 수 있도록 해독해주며 이후 ORB는 호출하려는 컴포넌트의 위치를 파악한뒤 요청을 전달하게 된다. 클라이언트는 ORB에서 일어나는 서버 객체와의 통신이나 활성화 방법, 그리고 객체의 위치, 작성 언어, 운영체제 등에 관해 전혀 알 필요가 없다. 이러한 ORB 기능을 이용하려면 IDL 명세를 따라 인터페이스를 구성하면 된다. CORBA 객체도 역시 인터페이스를 통해 의사소통을 하며 활성화된 객체는 객체 레퍼런스를 통해 식별되고 클라이언트는 이 객체 레퍼런스를 얻어 메소드 호출을 위한 핸들로 사용한다.

3.2 DCOM

DCOM은 소프트웨어 컴포넌트들이 네트워크를 통해 안정적이고 효과적인 방법으로 통신할 수 있는 프로토콜을 제공하는 분산객체 미들웨어이다. DCOM 환경에서 클라이언트는 원격 컴포넌트가 마치 자신의 주소공간에 있는 것으로 간주하고 인터페이스를 얻어 메소드를 호출하게 된다.

DCOM의 개념은 CORBA와 많은 면에서 비슷하다. 먼저, 각 서비스는 서비스 객체인 DCOM 서버에 의해 제공되며 DCOM 서버는 자신의 서비스를 하나 이상의 인터페이스를 사용해 외부에 공개한다. CORBA의 경우처럼 MIDL(Microsoft Interface Definition Language)을 통해 인터페이스를 정의한 후, MIDL 컴파일러를 사용해 프록시(proxy)와 스텁 코드를 생성한다.

DCOM 클라이언트는 프록시에서 제공되는 DCOM 서버 인터페이스를 통해 서버의 서비스에 접근하며 스텁은 DCOM 서버에서 사용된다. DCOM 서버는 제공하는 서비스에 대한 인터페이스 클래스를 생성함으로써 완성되며 클라이언트는 원하는 DCOM 서버에 접근하기 위해 사용하려는 인터페이스와 클래스를 알아야 한다.

DCOM에서는 이를 위해 고유한 GUID(Global Unique IDentifier)를 제공하는데 인터페이스에 대해서는 IID(Interface IDentifier)를, 클래스에 대해서는 CLSID(CLaSs IDentifier)를 제공한다.

DCOM 서버는 DLL 또는 EXE 화일로 만들어지며 DLL 형식으로 만들어진 DCOM 서버(in-proc 서버)는 클라이언트와 동일한 주소영역에서 동작하며 EXE 실행 화일로 만들어진 DCOM 서버(local 서버 또는 remote 서버)는 클라이언트와 분리된 주소 영역에서 동작할 수 있다.

CORBA와 DCOM 구조는 표 1에서 보는 것처럼 유사하며 둘다 원격 객체에 대한 투명한 접근과 활성화를 허용함으로써 분산객체 시스템의 하부 구조를 제공한다.

표 1 CORBA와 DCOM 비교

분산객체 미들웨어 / 비교항목	CORBA	DCOM
객체모델	전통적인 객체(object)	변형된 객체(component)
표준	OMA의 표준일부	인터페이스 이진표준
주관 기관	OMG	Microsoft
GUID지원(객체/인터페이스)	인터페이스이름	CLSID/IID
서버객체 탐색정보	implementation repository	registry
인터페이스 정보	interface repository	type library
객체구현을 찾는 모듈	ORB	SCM
마샬링지원(클라이언트/서버)	stub/skeleton	proxy/stub
언어지원	C, C++, Smalltalk,(Cobol, Java…)	C++, C
재사용기법	inheritance	aggregation, containment
서비스지원	다양한 서비스	제한된 서비스
지원플랫폼	Windows, Mac, OS/2, Unix, VMS	Windows(Unix)

4. ERP서버 기술

일반적으로 ERP서버는 전체 ERP시스템의 확장성(scalability), 이식성(portability), 상호운용성(interoperability) 및 개방성(openness) 그리고, 적용성(customizability) 등을 효율적으로 지원하여야 한다. 분산객체 컴퓨팅 환경하에서의 ERP서버를 위한 핵심 기술은 다음과 같이 분류할 수 있다.

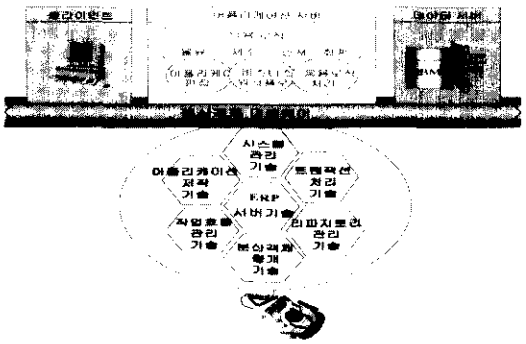


그림 2 ERP 서버 기술

4.1 시스템 관리기술

ERP시스템 관리 기술은 중간계층의 어플리케이션 서버에서 응용 로직(application logic)을 제외한 서비스 로직(service logic)으로 처리되어야 할 기능들을 포함한다. 여기에는 사용자 세션(session) 관리, 프로세스의 스케쥴링 및 부하 균등화(load balancing), 백그라운드 프로세스 처리 및 스풀링(spooling) 관리 및 ERP서버에 대한 모니터링 및 통제, 운영기능들이 포함된다.

4.2 어플리케이션 저작기술

업무분석을 통해 설계된 어플리케이션은 일관되고 통합된 개발환경을 통해 개발, 유지, 관리되어야 한다. 특히, 개발된 표준업무 어플리케이션은 다른 시스템 및 운영환경에서 재사용될 수 있도록 컴포넌트 형태로 저장, 관리될 필요가 있으므로 이를 효과적으로 지원할 수 있는 최적화된 저작 기술 및 저작도구가 필요하다. 여기에는 3계층 C/S구조하에서 효율적으로 실행되는 고유한 스크립트 프로그램 편집기, 다양한 화면 컨트롤을 지원하는 화면 편집기, 다양한 데이타베이스 테이블 구조를 정의하고 스키마를 생성할 수 있는 데이타베이스 편집기, 편리한 사용자 환경을 구성할 수 있도록 지원하는 메뉴 편집기 및 에러 분석을 위한 디버거 그리고, 편리한 사용자인터페이스를 지원하는 GUI 형태의 전용브라우저 등이 지원되어야 한다.

4.3 트랜잭션 처리기술

ERP시스템에서 모든 어플리케이션은 트랜잭션 지향적이며 따라서, 일반적인 트랜잭션으

로서 만족하여야하는 ACID 조건을 충족해야 한다. 그러나, ERP 트랜잭션은 업무의 실행과 논리적으로 밀접하게 연관된 다이알로그 스텝들의 연속으로 비즈니스 트랜잭션(business transaction)이라고 하는 트랜잭션이 발생된다. 비즈니스 트랜잭션이란 일반 데이타베이스 내에서의 트랜잭션보다는 보다 범위가 크며 논리적으로 연관된 업무단위에 포함된 여러 화면에 걸쳐 수행되는 트랜잭션으로 갱신(update) 연산을 주로 포함하며 장기 트랜잭션(long-duration transaction)의 특성을 갖는다. 이와 같은 ERP 비즈니스 트랜잭션을 처리하기 위해서는 데이타베이스 엔진과는 별도의 록 객체(lock object)와 트랜잭션 철회(abort)시 데이타의 일관성을 유지하기 위한 재실행 로그(redo log)방식의 회복 기법이 지원되어야 한다.

4.4 작업흐름 관리기술

ERP시스템은 네트워크 환경하에서 여러 사용자들에 의해 사용되게 되며, 작업중의 상당부분이 업무프로세스(business process)의 형태를 띄게 된다. 즉, 단일 어플리케이션이나 단일 사용자에 의해 작업이 완료되기 보다는 여러 사용자에 의해, 다양한 어플리케이션을 통해 하나의 업무가 완성되므로 이러한 작업흐름(workflow)을 총괄하여 작업프로세스를 관리하는 기술이 필요하다. 즉, 각 어플리케이션들은 미리 정해진 작업흐름을 따라 자동화된 흐름으로 사용자들에게 전달되어 수행되므로 작업흐름 관리는 ERP시스템 실행시에 중심적인 역할을 수행한다. 작업흐름을 관리하기 위해서는 사용자들이 자동화된 업무흐름을 쉽게 정의할 수 있도록 해주거나 혹은 표준화된 업무흐름을 기업목적에 맞게 수정할 수 있도록 하는 GUI 환경의 작업흐름 편집기(workflow editor)와 단순한 전자결재 수준이 아닌 복잡하고 다양한 업무 프로세스를 처리할 수 있는 작업흐름 엔진(workflow engine)이 지원되어야 한다.

4.5 분산객체 중개기술

분산객체 컴퓨팅 환경에서 이질적인 운영환경에 위치한 서버내의 컴포넌트들을 객체 개념을 기반으로 하면서도 쉽고 일관된 방식으로 접근할 수 있도록 중간에서 중개하는 것은 가장 기본적인 기술이다. 이와 관련한 기본 기능들은 대표적인 분산객체 미들웨어인 CORBA와 DCOM이 대부분 지원하기때문에 이에 대한 활용기술이 요구된다. CORBA 혹은 DCOM 중에서 개발 또는 적용하려는 ERP시스템 운영환경에 따라 적절한 중개자(broker)를 선택하고 IDL 컴파일러를 통해 클라이언트와 서버측 통신기능을 제공하는 코드를 자동적으로 얻을 수 있으므로 호출하고자 하는 적절한 서버측의 실행 인터페이스를 정의하고 구현하는 것이 필요하다.

4.6 리파지토리 관리기술

리파지토리(repository)는 기업의 경영전략에서 구현까지의 모든 정보를 통합적으로 관리하여 유연성, 포괄성, 개방성을 제공하는 개발환경에 독립적인 메타정보 관리기법이다. 이러한 리파지토리는 ERP와 같은 패키지 방식인 개발도구에 활용함으로써, 전사적 관점에서 기업의 자원과 지식의 효과적 통합 관리가 가능하다. 실제로 SAP(Systems, Applications and Products in data processing)과 같은 선진 ERP시스템에서는 업무 개념을 객체지향적으로 모듈화한 비즈니스 객체가 리파지토리에 저장되는 방식을 통해 특정 업무영역의 표준지식을 표현하고 있다. 리파지토리의 장점은 특정 개발환경에 독립적으로 적용가능하고 이질적인 개발 도구들을 상호유기적으로 연결하여 통합적인 개발접근이 가능하며, 장기간 사용시 기업정보의 축적이 가능해진다는 것이다. 궁극적으로 리파지토리는 조직의 모든 지식자산을 통합적으로 저장관리할 수 있는 기반기술로 활용될 수 있다.

리파지토리는 전체 정보시스템의 구현 및 관리의 모든 분야를 포괄하는 개념이기 때문에 개발과 관련한 자원관리 부분뿐만 아니라 개체-관계도의 개체와 속성 그리고 개체간의 관계, 비즈니스 규칙, 자료개체 타입, 테이블 정의, 프로그램 소스코드, 화면, 개발관련 서류, 도움말, 아이콘 등 시스템 개발에 필요한 모든 자료가 포함되어야 한다.

5. ERP서버 구축예

5.1 운영 환경

ERP서버 구축예로는 ETRI에서 개발한 3계층 C/S구조에서의 '표준정보시스템'에 대해 설명한다. 개발한 표준정보시스템은 표준업무 처리를 위한 비지니스 어플리케이션에 대한 컴포넌트 개념을 지원하고 독자적인 개발툴과 실행 브라우저를 통한 어플리케이션 개발과 실행이 가능한 DCOM 분산객체 미들웨어 기반의 ERP 시스템이다. 개발한 표준정보시스템의 운영 환경은 그림 3과 같다.

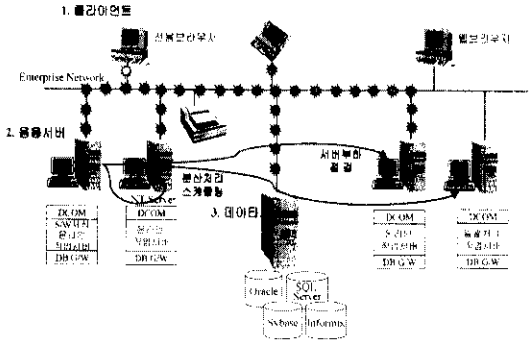


그림 3 ERP 서버의 운영 환경

개발한 표준정보시스템은 현재는 3계층 환경에서 실행중이며 n계층 환경으로의 확장이 진행중이다.

5.2 표준정보시스템 구조

표준정보시스템은 그림 4와 같은 3계층 C/S 구조를 갖는다. 특히, 응용 로직을 처리하는 어플리케이션 서버는 응용 로직을 처리하기 위한 다양한 인터페이스들을 제공하는 DCOM 오브젝트들로 구성되어 컴포넌트 계층(component-tier)을 형성하며 MTS(Microsoft Transaction Server)에 의해 패키지 단위로 관리되면서 분산 2단계 완료 규약(2 phase commit protocol)과 자원 풀링(resource pooling)을 지원받는다.

● 1계층 : 표현 계층

클라이언트는 다양한 저작 도구(프로그램 작성기, 화면 작성기, 데이타베이스 작성기, 메뉴

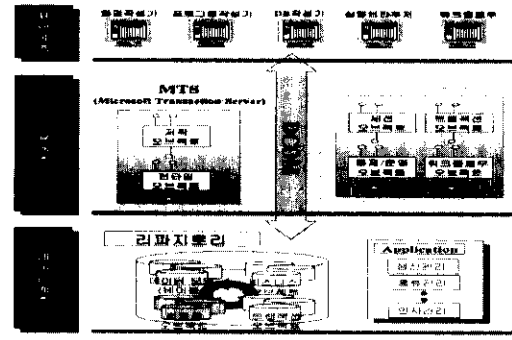


그림 4 표준정보시스템 구조

작성기, 객체 브라우저 등)와 실행 브라우저, 그리고 작업흐름 관리기 등을 수행함으로써 사용자에게 인터페이스를 제공하며 사용자의 요청을 받고, 그 결과를 사용자에게 되돌린다.

● 2계층 : 컴포넌트 계층

ERP서버 역할을 하는 중간계층인 컴포넌트 계층은 런타임 오브젝트, 저작 오브젝트, 세션 오브젝트, 트랜잭션 오브젝트, 워크플로우 오브젝트, 서버통제/운영 오브젝트 등의 여러 COM 오브젝트들로 구성되어 있다.

리파지토리로부터 비즈니스 컴포넌트를 꺼내어 응용 로직을 처리하는 런타임 오브젝트는 스크립트엔진을 통해 응용 로직 스크립트를 인터프리터 방식으로 수행하는데 이를 위한 메소드들을 별도의 레코드셋과 스크립트 처리를 위한 인터페이스를 통해 제공한다. 모든 COM 오브젝트는 QueryInterface( ), AddRef( ), Release( ) 메소드를 갖는 IUnknown이라는 인터페이스를 기본적으로 제공하며 다른 인터페이스들을 통해 고유한 기능들을 지원한다. 저작 오브젝트는 프로그램 저작을 위한 IProgram, 화면 저작을 위한 IScreen, 데이타베이스 스키마 정의를 위한 IDatabase, 메뉴저작을 위한 IMenu, 객체들을 관리하고 트리뷰를 제공하는 IObject 등의 인터페이스를 제공한다. 이외에도 트랜잭션, 워크플로우, 서버통제/운영 오브젝트들이 제공하는 많은 인터페이스 정보들은 트리형태의 윈도우 레지스트리(registry)에 등록되어 클라이언트쪽에서 원격으로 DCOM을 통해 원하는 인터페이스를 지원하는 COM 오브젝트를 조회하고 필요한 메소드를 호출할 수 있도록 되어 있다.

컴포넌트 계층에 존재하는 오브젝트들은 MTS라고 하는 또다른 미들웨어에 의해 패키지단위로 내포되어 MTS가 제공하는 분산 트랜잭션, 쓰레드 관리, 자원 공유 기능들을 사용한다.

● 3계층 : 데이타 계층

데이타 서버는 기업내의 생산, 자재, 영업, 인사, 회계 등의 표준화된 업무 로직을 구현한 표준 비즈니스 어플리케이션을 컴포넌트 형태로 IRDS 모델을 참조하여 구성한 리파지토리 안에 저장, 관리한다. 이 리파지토리 안에는 프로그램, 화면, 매소드, 데이타테이블 등과 같은 기본 컴포넌트이외에도 개발 문서, 메뉴얼 등의 컴포넌트들이 저장된다.

6. 결 론

본 논문에서는 분산객체 컴퓨팅 환경하에서의 ERP서버 기술에 대해 기술하였다. 국내외의 ERP시스템들의 기술 현황과 특성을 분석한 결과, 좀더 유연하고 개방적인 분산컴퓨팅 환경을 수용하는 ERP 시스템 기술이 필수적임을 알 수 있었다. 특히, 이질적인 운영환경에 대한 투명성을 제공하는 분산객체 미들웨어를 활용한 n계층 ERP서버는 전체 시스템의 성능과 유연성을 결정하는 ERP의 핵심이다.

또한, 본 논문에서는 이를 지원하는 DCOM과 CORBA를 활용한 ERP서버 구축의 핵심기술과 그 적용 사례를 살펴보았다. 분산객체 컴퓨팅 환경하에서의 ERP서버는 서버자체에 대한 관리 및 통제, 어플리케이션 저작, 트랜잭션 처리, 작업흐름 관리, 분산객체 중개 및 리파지토리 관리 기술들이 요구되는 어렵지만 매우 중요한 기술이다. 개발된 표준정보시스템은 컴포넌트 개념의 표준 어플리케이션 로직을 리파지토리에 저장, 관리함으로써 재사용과 적합화를 가능케하며 고유한 개발도구와 실행 브라우저를 함께 지원하는 DCOM기반의 ERP 전용 서버이다.

ERP시스템은 기업내의 정보인프라 구축의 핵심이며 앞으로 다가올 CALS, EC 환경을 위한 기본 토대가 될 것으로 예측된다. 그결과 ERP서버 역시 그 중요성이 점차 증가할 것이며 분산객체 미들웨어와 같은 최신의 컴퓨팅 기술을 활용한 지속적인 서버의 확장이 요구된다.

참고문헌

[1] Orfali, Harkey, The Essential Client/Server Survival Guide, Wiley, 1996.

[2] Orfali, Harkey, Client/Server Programming whit Java and Wiley, 1997.

[3] Thomas J. Mowbray, Ron Zahavi, The Essential CORBA System Integration Using Distributed Objects, John Wiley & Sons Inc., 1995.

[4] Jon Siegel, CORBA Fundamentals and Programming, Wiley, 1996.

[5] Paul E. Renaud, Introduction to Client/Server Systems, Wiley, 1993.

[6] Guy Eddon, Henry Eddon, Inside Distributed COM, Micresoft Press, 1998.

[7] Roser Jennings, Database Workshop - Microsoft Transaction Server 2.0, SAMS Publishing, 1998.

[8] Yen. p. Shan, Ralph H. Earle, Enterprise Computing with Objects from Client/Server to the Internet, Addison Wesley, 1998.

[9] Don Box, Essential COM, Addison Wesley, 1998.

[10] Thomas J. Mowbray, William A. Ruh, Inside CORBA-Distributed Object Standards and Applications, Addison Wesley, 1998.

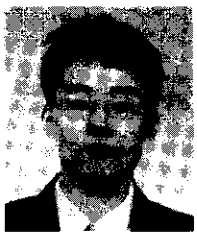
[11] Rudiger K, Wolfgang Weiss, ABAP/4 SAP'S R/3 Applications Developing, 삼각형.

[12] Orfali, Harkey and Edwards, The Essential Distributed Object Survival Guide, Wiley, 1996.

[13] Geofine A., "The Information Resource Dictionary System", Proc. of the 15th International Entity-Relationship Conference, 1985.

[14] Lefkovits and Sibley, "Information Re-

source / Dictionary Systems", Wellesley, 1983.
[15] Daniel R. Dolk, "Model Management and Structured Modelling : The Role of Information Dictionary System", Communication of the ACM, Vol. 31, No. 6, 1988.
[16] Bodoloi, B., sIRCAR, s., and Lakhanpal, B, "Desirable Characteristics of Information Resource Dictionary Systems", Journal of Database Management, 1998.
[17] Asrafi, n and Kuilboer, J., "The Information Repository : A Tool for Metadata Management", Journal of Database Management, 1995.
[18] 이교상, 박화규, 손주찬, 고영철, 백종명, 박상봉, "중소기업형 표준정보시스템 개발에 관한 연구," 한국CALS/EC학회지, pp. 137-154, 1997. 12.
[19] 백종명, "표준정보시스템(ERP) 기술 개발 개요," '98 CALS/EC 기술개발 종합 세미나 발표집, pp. 229-234, 1998. 5.



박 성 진

1991 고려대학교 전산과학과(이학사)
1993 고려대학교 전산과학과(이학석사)
1998 고려대학교 전산과학과(이학박사)
1998～현재 한국전자통신연구원 정보통합연구팀 선임연구원
관심분야 : ERP 및 CALS, 분산 및 통합 데이타베이스, 분산객체 컴퓨팅

E-mail : sjp@etri.re.kr

문 봉 교

1992 서강대학교 전산학과(공학사)
1992～1996 (주)대외반도체연구소 주임연구원
1998 광주과학기술원 정보통신공학과(석사)
1998～현재 한국전자통신연구원 정보통합팀 연구원
관심분야 : ERP 및 CALS, 분산컴퓨팅, 이동컴퓨팅, 통신프로토콜

E-mail : bkmoon@etri.re.kr