CORBA 환경하의 CALS 통합 데이터베이스 설계

우훈식, 정석찬, 윤선희, 조장혁, 주경준

시스템공학연구소 CALS연구실

요약

CALS 통합 데이터베이스는 지역적으로 분산되어 있고 이질적인 형태를 갖는 기술 및 관리 정보를 디지털 형태로 변환하여 교환 및 공유할 수 있도록 하는 정보 공유 체제로 CALS 구현의 핵심 정보 기술이다. 또한, 분산된 CALS 정보는 체계적으로 통합되어 글로벌 뷰를 제공하여야 하므로 이질의 데이터베이스를 통합하는 것이 매우 중요하다. 본 논문에서는 CALS 통합 데이터베이스의 주요 핵심 기술을 고찰하고, 이를 분산 객체 환경인 CORBA 하에서 구현하기 위한 시스템 모형을 제시한다.

1. 서론

CALS (Continuous Acquisition and Life cycle Support 또는 Commerce At Light Speed)는 미국 국방부의 주도로 냉전 체제 이후 격감된 국방비 하에서 일정 수준의 군사력을 유지하려는 의도로 시작되었으며, 미군의 무기 체계에 대한 기획, 설계, 제조, 군수, 폐기 등의 라이프 사이클 관련 정보를 표준화하여 이를 관련 정부 및 군수 기업들 간에 상호 교환 및 공유하여 비용을 절감하자는 목적을 가지고 있다 [12]. 지난 10여년간의 효과적인 CALS 추진으로 현재의 CALS는 군수 산업뿐만 아니라 제조업, 유통, 건설 등의 전 산업에 걸친 산업 정보화의 핵심 용어로 사용되고 있다 [15]. 이러한 CALS를 산업에 구축하는 과정 중 가장 큰 어려움 중의 하나는 네트워크 상에서 분산되어 있는 이질적인 CALS 라이프사이클 정보에 대한 관리를 효율적으로 제공하기 위한 시스템에 관한 문제이다 [16].

구축된 CALS 시스템은 이질 분산 환경하의 정보를 효율적이고 통합적인 방법으로 교환 및 공유할 수 있도록 하여야 한다. 여기에서 통합이란 의미는 물리적으로 하나의 정보 저장소를 의미하는 것이 아니라 논리적인 데이터 통합을 말하는 것으로 다양한 형태의 정보를 언제 어디에서나 투명하게 실시간에 접근할 수 있도록 하는 체제를 의미한다.

또한, CALS 구현을 위해서는 이질성을 고려하여 분석하는 것이 매우 필요하다. CALS 환경하의 이질성은 플랫폼 수준, 데이터베이스 관리 시스템 수준, 그리고 시멘틱 수준의 이질성이 있다. 플랫폼 수준에서의 이질성은 시스템이 상이한 하드웨어상에 존재하며 상이한 운영 체제에 의해 관리되고 서로 다른 통신 프로토콜을 사용하는 것에서 기인하며, 데이터베이스 관리 시스템 수준에서의 이질성은 데이터를 관리하는 관리 시스템의 종류 즉 파일, 관계형, 그리고 객체형 관리 시스템의 차이에 기인한다. 또한, 시멘틱 이질성은 서로 다른 데이터베이스가 독립적으로 설계되었기 때문에 발생하는 것으로 스키마 충돌과 데이터 충돌을 포함한다.

이질 분산 환경하의 네트워크 컴퓨팅에 대한 최근의 추세는 분산 객체 컴퓨팅 기술을 이용하는 것이다 [11]. 즉, 통신의 인프라로 분산 객체 컴퓨팅 표준인 CORBA 환경을 이용하여 이질적인 복수의 서버를 연결시키셔 클라이언트에 대한 투명성 있는 액세스를 제공할 수 있도록 하는 것이다 [16]. 따라서, 본 연구에서는 CORBA 를 이질 객체 간의 통신을 위한 기본 구조로 사용한다.

CORBA 는 이기종간의 시스템 상에 있는 객체들을 독립적인 접근을 구현하도록 제공하는 명세서 및 아키텍처이므로 일차적으로 플랫폼 레벨에서의 이질성을 제어한다. 또한, CORBA 를 사용하므로써, 분산된 객체와 관련된 오퍼레이션들의 집합으로 애플리케이션을 캡슐화하는 것이 가능해지므로 클라이언트와 서버의 기능을 Plug-In 혹은 Plug-Out 할 수 있으므로 데이터베이스의 추가나 혹은 재배치가 매우 쉽게 된다. 따라서, 데이터베이스 수준에서의 이질성이 제어된다. 스키마 충돌과 같은 시멘틱 이질성은 애플리케이션 프로그래밍 레벨에서 해결할 수 있으며 본 연구에서는 구현 프로토타입을 제안한다.

2. 분산 시스템

분산 시스템 개념이 도입되기 전에는 중앙에 있는 메인 컴퓨터에 단말기를 통하여 작업을 수행하는 호스트 중심의 중앙 집중식 방법이 주류를 이루었다. 이 방법은 중앙의 호스트가 모든 작업을 수행하므로 작업량이 많아질수록 호스트의 부하가 문제가 되어서 시스템의 성능 증대가 필요하였고 따라서 가격이 계속 상승하는 문제를 동반하였다 [7].

이와 같은 문제를 해결하고자 제안된 방법이 분산 시스템으로 기본 개념은 작업을 일정하게 분배하여 수행하게 하는 것으로 UNIX 를 기반으로 하는 시스템이 주종을 이루었으며 초창기 대표적인 기술은 클라이언트/서버 모형이다 [2].

초기의 클라이언트/서버 모형은 2-Tier 모형으로 원격 데이터베이스 (Database, DB) 접근 모델과 데이터베이스 서버 모델로 나눌 수 있다 [2,12,14]. 원격 DB 접근 모델은 클라이언트 측에 SQL (Structured Query Language)문을 포함하는 응용 프로그램을 탑재하고 서버는 단지 데이터베이스만을 관리하는 시스템이다. 이 시스템은 기존의 메인 호스트 중심 시스템의 한계를 어느 정도 극복할 수는 있었으나, 응용 프로그램이 데이터베이스 관리 시스템 등의 시스템 구성 요소에 종속적으로 작성되다 보니 다른 시스템으로의 이식성, 그리고 확장성 등의 문제가 제기 되었다. 이러한 문제점에도 불구하고 현재 구축된 대부분의 클라이언트/서버 시스템은 원격 DB 접근 모델을 따르고 있다.

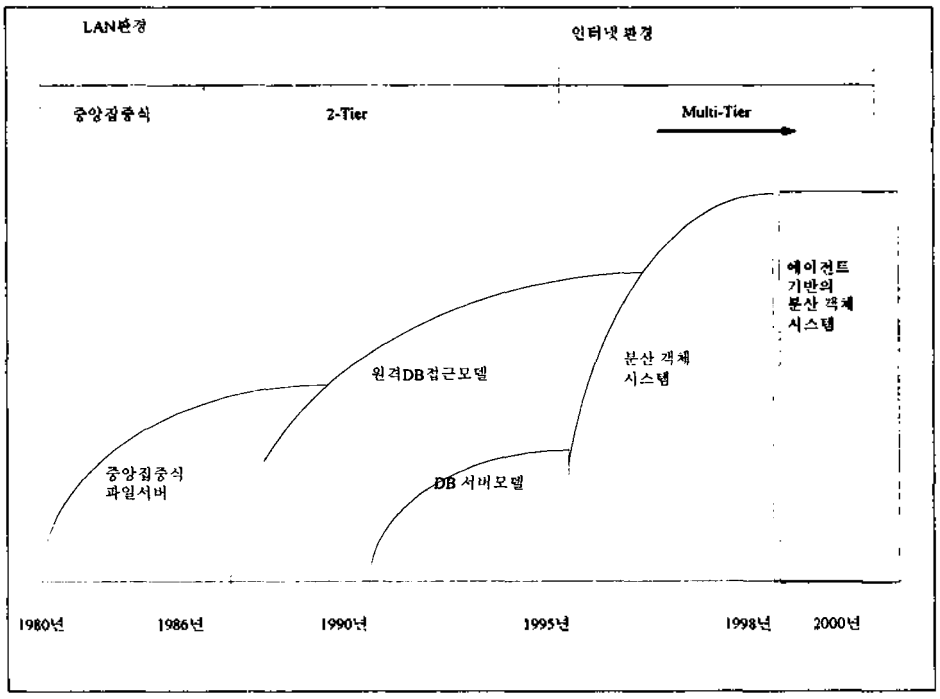
DB 서버 모델은 원격 DB 접근 모델의 이식성/확장성 한계를 보완하기 위한 모델로 데이터베이스 관리 시스템에서 지원하는 저장 프로시쥬 (Stored Procedure)를 사용하여 응용 프로그램을 작성하는 것이다. 시스템의 로직에 변화가 있는 경우에 저장 프로시쥬를 변경하여 유지 보수를 쉽게 하도록 하는 장점을 가지고 있다.

이와 같은 초기 2-Tier 클라이언트 시스템은 LAN 환경에서는 적절하게 사용되었지만, WAN 종류의 네트워크 상에 분산된 이질적인 데이터베이스를 계속적으로 연동시켜 통합하는 환경 구현에는 많은 문제점이 발견되었다. 따라서, 클라이언트와 서버, 그리고 응용 프로

그램을 완전히 분리하는 시스템이 제안되었으며, 이것이 최근에 개념이 소개되는 3-Tier 클라이언트/서버 모델이다 [11]. 3-Tier 클라이언트/서버 모델은 클라이언트, 서버, 그리고 클라이언트와 서버의 상호 연동을 위한 미들웨어 등 세가지 주요 요소로 구성된다. 클라이언트 요소는 구현 시스템의 클라이언트 부분을 수행하는 것으로, 분산된 서비스를 요청하는 GUI (Graphical User Interface)의 형태로 구현된다. 서버 요소는 구현 시스템의 서버 부분을 수행하는 것으로 응용 분야에 따라 데이터베이스 서버, TPM (Transaction Processing Monitor), 그룹웨어 서버, 객체 서버, 또는 웹 서버의 형태로 구현된다. 미들웨어 요소는 클라이언트와 서버 사이의 상호 연동을 지원하기 위한 소프트웨어로서 통신 미들웨어, 데이터베이스 미들웨어, OLTP (On Line Transaction Processing) 미들웨어, 객체 미들웨어 등으로 구성되며 DCE, TCP/IP, HTTP, ODBC, DCOM, CORBA 등이 현재 사용되고 있다. [그림 1]은 LAN 환경하의 중앙 집중식 시스템에서 2-Tier 시스템 그리고 글로벌 환경하의 Multi-Tier 시스템에 이르는 분산 시스템의 기술적 시대 변화 과정을 보여 준다.

3. CORBA

CORBA (Common Object Request Broker Architecture)는 1989 년 분산 환경하에서 객체 기술을 바탕으로 응용 프로그램을 결합하기 위해 구성된 OMG (Object Management Group)에 의해 제안된 OMA (Object Management Architecture)에 기반한다 [9,11]. OMA 는 응용 프로그램의 단순한 결합에서 벗어나 객체의 생성, 소멸, 저장, 트랜잭션 등의 분산 객체 환경에서 필요한 모든 기능을 정의하였다. 즉, Object 간의 요청과 응답에 대한 송수신을 가능하게 하는 메시



[그림 1] 분산 시스템 기술 변화 과정

징을 담당하는 객체 요청 중개자인 ORB (Object Request Broker)와 객체 시스템을 실현하고 유지하기 위한 서비스의 집합인 공통 객체 서비스 (Common Object Service Specification; COSS), 애플리케이션이 사용되는 범용 기능을 제공하는 클래스 객체의 집합인 공통 지원 기능(Common Facility), 그리고 사용자 애플리케이션 자체를 나타내는 응용 객체 (Application Object)로 구성된다 [9,10].

CORBA 는 OMA 중에서 객체 간의 통신을 담당하는 ORB 의 사양이다. 즉, 클라이언트/서버 개념으로 클라이언트 객체가 발행하는 요청을 해당 서버 객체에 송신하고 그 결과를 다시 클라이언트 객체에 반환하는 기능을 정의한 것이다. CORBA 는 1991 년 최초로 사양이 정의된 이래 현재 CORBA 2.0 사양까지 발표되었으며 최근에는 CORBA 3.0 이 준비중이다.

이기종 객체 간의 통신에서 중심적인 역할을 하는 것은 OMA 구성 요소 중 객체 요청 중개자인 ORB 이다. 즉, 각 객체를 정의하고 이를 객체 요청 중개자에 등록하면 서로 다른 언어나 환경에서도 객체 인터페이스를 통하여 통신할 수 있으며, 이 때 객체 요청 중개자는 응용 객체와 객체 요청 중개자 사이에 있는 Proxy 를 이용하여 클라이언트로부터 전달된 메시지를 등록된 객체에 전송한다. ORB 는 [그림 3]과 같이 구성되며 각 구성 요소는 아래와 같이 정의된다 [10].

- ORB CORE

  클라인언트/서버 객체간의 상호 연결을 가능하게 해주는 핵심 요소로 객체간의 상호 연결하는 버스로 객체간 통신을 위한 투명성을 보장한다.

- OMG IDL (Interface Definition Language)

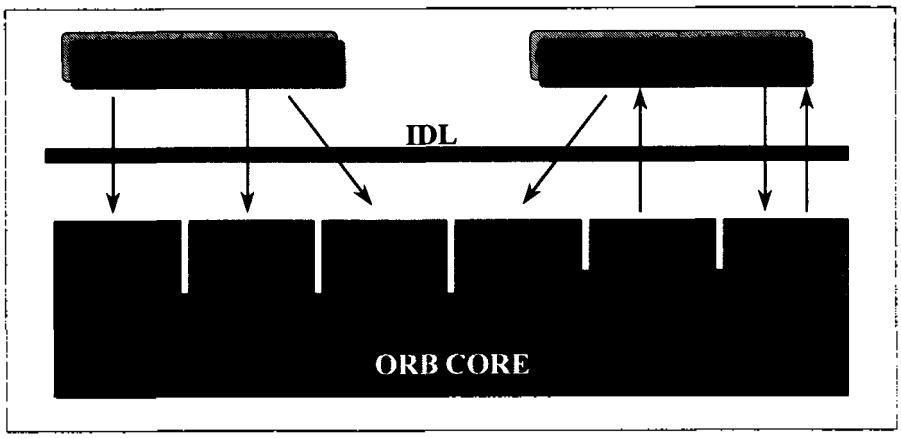
  객체의 인터페이스와 데이터 구조를 정의하기 위한 정의 언어

- Client Stubs

  클라이언트 쪽의 객체에 대한 정적 인터페이스를 제공

- Dynamic Invocation Interface (DII)

  실행 시간에 객체를 호출할 수 있는 동적 인터페이스를 제공



[그림 2] ORB 구성 요소

- IDL Skeleton

  Object Implementation 쪽에 구현된 특정 객체를 정적으로 호출하기 위한 기능
- Implementation Repository

  실행 서버쪽에 구현된 객체에 관한 정보를 저장하여 객체가 호출될 때 이용 가능하게 해주는 기능
- Dynamic Skeleton Interface

  서버쪽에 구현된 특정 객체를 동적으로 호출하기 위한 기능
- Object Adapters

  ORB core 가 객체 구현 부분에 접근할 수 있게 해주는 기능
- ORB Interface

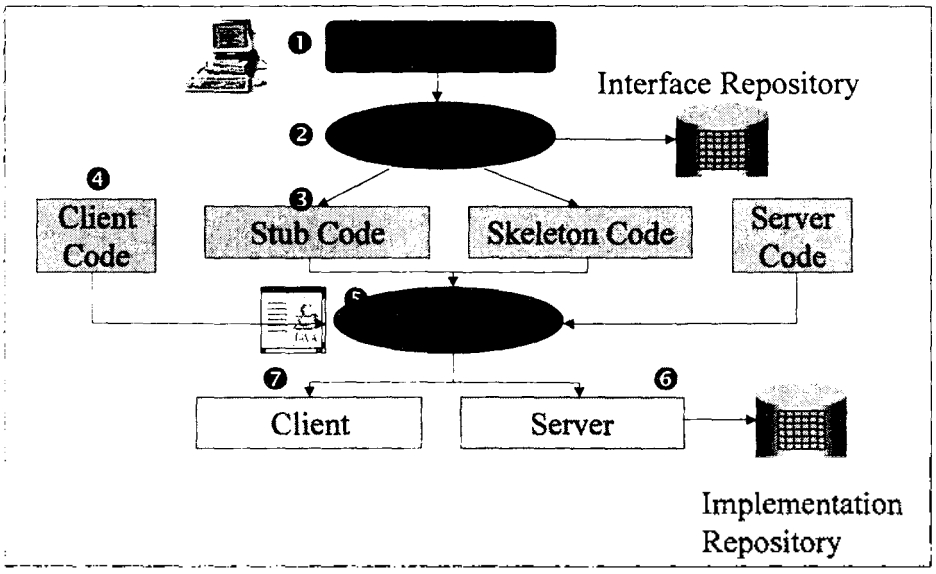
  ORB Core 가 제공하는 서비스를 호출하는 인터페이스를 제공
- Interface Repository (IR)

  IDL 에서 정의한 모든 인터페이스, 메소드, 상수, 타입 선언 등을 저장 관리하여 클라이언트가 요청한 서비스에 대하여 인터페이스 정보를 조회하여 검사한다.

이와 같은 ORB 구성 요소와 CORBA 미케니즘을 이용하여 분산 클라이언트/서버 프로그램을 작성하는 과정은 [그림 3]과 같으며 작성 단계는 다음과 같다.

단계 1. OMG IDL을 이용하여 객체 인터페이스 정의

단계 2. Vendor 제공의 IDL 컴파일러를 이용하여 IDL Stub과 IDL Skeleton을 목표로 하는 언어로 매핑 및 생성



[그림 3] CORBA 분산 객체 클라이언트/서버 프로그램 작성 과정

단계 3. Stub 코드와 Skeleton 코드 작성
단계 4. 클라이언트와 서버 코드를 해당 프로그래밍 언어로 작성
단계 5. 단계 3과 단계 4의 코드를 컴파일
단계 6. 서버 부분을 저장소에 저장
단계 7. 클라이언트 프로그램 수행

4. CALS 통합 데이터베이스 시스템

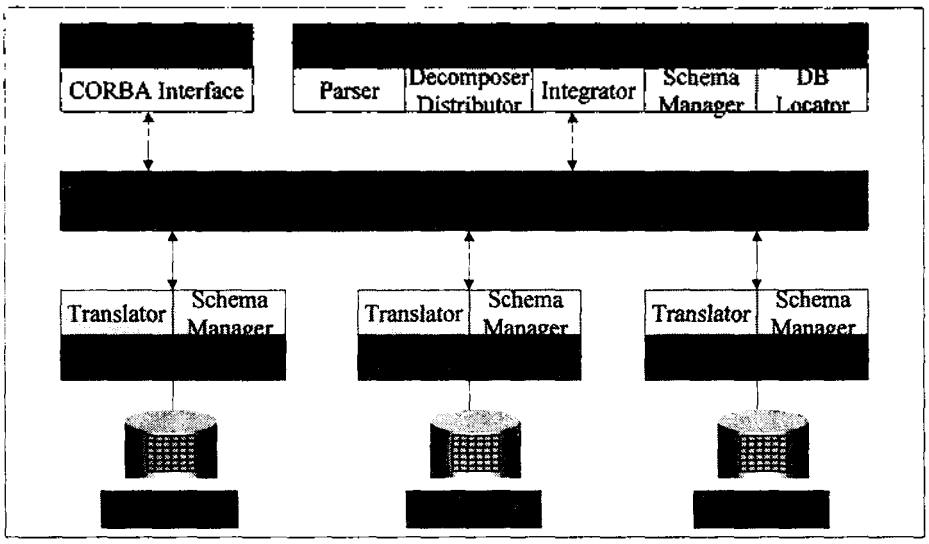
CALS 통합 데이터베이스는 정부 및 업계의 제품 또는 체제의 라이프 사이클에 공통 데이터 환경을 지원하는 논리적 데이터베이스로 기술 및 비기술 데이터를 보유하는 물리적인 데이터베이스로부터 생성된다 [6,16]. CALS 통합 데이터베이스에서 제공하는 공통 데이터 환경은 비통합 데이터베이스에서 흔히 볼 수 있는 데이터의 중복과 불일치를 제거하는 것이 주 목적이며 데이터를 한번 생성하여 여러 번 사용하는 CALS의 기본 목적을 충족시키는 필수적 요소이다 [5].

CALS 통합 데이터베이스 시스템은 분산된 이질 시스템의 모든 데이터를 논리적인 뷰로 표현하는 글로벌 스키마를 생성하는 것이 기본 개념이며, 주요 기능은 다음과 같다 [5].

1) 분산 투명성
   사용자는 자신의 로컬 데이터베이스에 접근하는 것처럼, 다수의 이질 데이터베이스에 접근할 수 있다.
2) 이질 투명성
   사용자는 자신의 로컬 데이터베이스에서 사용했던 모델과 언어를 사용하는 것과 같이 다른 데이터베이스의 스키마를 접근할 수 있다.
3) 규모성
   최소의 변화로 시스템은 네트워크를 통한 새로운 데이터베이스를 첨가하거나 기존의 데이터베이스를 제외할 수 있다.
4) 로컬 자치성
   CALS 통합 데이터베이스 하에서 로컬 사이트는 데이터베이스의 정보 내용, 데이터 모델 및 저장 구조에 대한 설계와 조회, 저장, 갱신 등의 로컬 운용에 대한 자율성을 갖는다.

5. 프로토타입 설계

본 장에서는 글로벌 환경하의 CALS 구현을 위하여 네트워크에 분산되어 있는 여러 이질 데이터베이스 서버를 OMG의 CORBA 환경하에서 통합하고 이를 클라이언트를 이용하여 질의를 수행할 수 있는 CALS IDB 시스템을 제안한다. 제안된 CALS IDB에서는 ORB를 미들웨어로 사용하는 CORBA 기반의 3-Tier 구조를 채택하였다. [그림 4]는 제안된 시스템의 구조를 나타내며, GUI를 표현하는 클라이언트, 글로벌 스키마를 관리하고 사용자 질의를 처리하는 글로벌 DB 컨트롤러, 그리고 각각의 로컬 DBMS에 의해 제공되는 추출 스키마를 관리



[그림 4] CALS IDB 프로토타입

하고 글로벌 컨트롤러에 질의 결과를 전달하는 로컬 DB 컨트롤러등으로 구성된다.

5-1. 클라이언트

본 프로토타입의 사용자 클라이언트는 ORB를 지원하는 클라이언트이다. 서버인 글로벌 DB 컨트롤러와의 통신은 ORB와 연동하며 송수신한다. 대화형의 글로벌 스키마 조회, 질의 송수신과 같은 서비스를 윈도우 형식으로 표현할 수 있도록 한다.

5-2. 글로벌 DB 컨트롤러

글로벌 DB 컨트롤러는 클라이언트에 대하여는 서버 역할을 그리고 로컬 DB 컨트롤러에 대해서는 클라이언트의 역할을 수행한다. 글로벌 DB 컨트롤러는 CORBA 객체로 글로벌 스키마의 정의를 관리하며 사용자의 질의에 대한 처리를 담당한다. 즉, 사용자와 분산 데이터베이스 간의 중재자의 역할을 수행하는 것이다. 클라이언트 객체를 통하여 사용자로부터 질의를 받으면 글로벌 DB 컨트롤러는 이를 정규 데이터 모델의 글로벌 스키마에 대한 질의로 해석한다. 이렇게 번역된 질의를 해당되는 로컬 데이터베이스 별로 분해하여 로컬 데이터베이스가 이해할 수 있는 언어로 다시 번역하여 각각의 로컬 데이터베이스가 처리할 수 있도록 질의를 발송하는 것이다. 발송된 질의는 로컬 DB 컨트롤러를 통하여 질의 결과가 수집되며, 이렇게 수집된 결과를 다시 사용자가 볼 수 있도록 통합하여 일치된 형태로 사용자가 볼 수 있도록 표현한다.

글로벌 DB 컨트롤러의 주요 구성 객체는 다음과 같다.

1) Query Parser

사용자 질의어를 파싱

2) Decomposer/distributor

글로벌 질의를 로컬 DB 별로 분리하여 해당 로컬 DB 에 발송

3) Integrator

로컬 DB 로부터의 질의 결과를 통합

4) Schema information manager

글로벌 스키마 정의 및 구성에 대한 관리 및 스키마 통합 작업 수행

5) DB location manager

글로벌 스키마 구성에 참여하는 로컬 DB 의 위치 정보를 수록/관리

5-3. 로컬 DB 컨트롤러

로컬 DB 컨트롤러는 글로벌 DB 컨트롤러로부터 글로벌 질의를 받아서 해당 로컬 DBMS 의 질의어로 번역하여 질의를 수행하여 그 결과를 글로벌 컨트롤러에 되돌려 주는 CORBA 객체로 지원하는 DBMS 에 따라 다르게 구현된다. 본 연구에서는 관계형 DBMS 인 ORACLE 과 SYBASE, 그리고 객체지향형 DBMS 인 ObjectStrore 를 고려하고 있다. 또한, 로컬 DB 컨트롤러는 기존에 운영되는 DB 서버의 자율성을 침해하지 않기 위해 글로벌 스키마에 참여하는 추출 스키마 만을 관리한다.

로컬 DB 컨트롤러의 주요 구성 요소는 다음과 같다.

1) Translator

글로벌 질의를 로컬 DBMS 의 질의어로 번역

2) Export schema manager

글로벌 스키마에 참여하는 추출 스키마를 관리

5-4. 스키마 통합

스키마 통합은 로컬 DB 의 이질적인 데이터 모델에 의한 스키마의 이질성을 해소하기 위하여 글로벌 스키마를 생성하는 것이다 [1,13]. 스키마 통합에서는 여러 데이터 모델을 연결하는 정규 데이터 모델의 사용이 요구되며, 그 외에도 이름 충돌, 표현 충돌 등을 해결하는 것이 중요하다 [8]. 이렇게 통합 데이터 모델에 의해 정의된 글로벌 스키마는 다음의 특성을 갖아야 한다 [5].

1) 완전성 및 정확성

글로벌 스키마는 로컬 스키마에서 추출된 스키마를 정확하게 표현하여야 한다.

2) 최소성

글로벌 스키마는 중복되지 않도록 그 수를 최소화하여야 한다.

3) 이해성

글로벌 스키마는 사용자와 개발자가 모두 쉽게 이해할 수 있도록 표현되어야 한다.

본 연구에서 고려된 스키마 통합 과정은 다음과 같다.

1) 참여 스키마 결정

참여 스키마 결정 단계는 스키마 통합에 대한 로컬 스키마의 참여 범위를 결정하는

것으로 로컬 DB에서 통합에 필요한 스키마를 결정하여 참여 스키마를 생성한다.

2) 스키마 치환

스키마 치환 단계는 참여 스키마 결정 단계에서 생성된 로컬 스키마를 정규 데이터 모델의 부분 집합 형태 즉 정규 모델 로컬 스키마로 치환하는 것이다. 정규 데이터 모델은 전술한 바와 같이 로컬 DB에서 추출된 스키마를 모두 수용할 수 있어야 하며 최근의 연구 추세는 객체 지향형 모델을 사용하는 것이다 [1,3,13].

3) 추출 스키마 정의

이 단계에서는 스키마 치환 단계의 정규 모델 로컬 스키마의 일부로 글로벌 스키마에 참여하는 추출 스키마를 결정하는 것이다. 결정된 추출 스키마는 로컬 DB 컨트롤러의 Export Schema Manager에 의해 관리 된다.

4) 스키마 통합

스키마 통합 단계에서는 스키마 간의 충돌을 발견하고 이를 해소하여 글로벌 스키마를 생성하는 것이다. 이렇게 생성된 글로벌 스키마는 Schema Information Manager에 의해 관리된다.

6. 결론

분산 환경하의 CALS 시스템에서는 공간적으로 분산되어 있는 기술 및 관리 자료가 디지털 형태로 변환되어 논리적으로 통합되어야 하므로 이질 정보를 효율적이고 통합적인 방법으로 교환 및 공유할 수 있는 체제가 인터넷과 같은 글로벌 네트워크에서 사용되어야 한다. 이렇게 이질 분산 환경하에서 CALS 시스템을 구축하는 것은 많은 기술적인 어려움이 있으며, 새로운 대안은 분산 객체 컴퓨팅 기술인 CORBA를 이용하는 것이다.

본 연구에서는 이질 분산 환경하의 CALS 구현을 위하여 상호 운용성, 위치 투명성, 및 확장성을 증가시키기 위해 CORBA 기반의 CALS IDB 시스템 모형을 제안하였다. 제안된 CALS IDB는 사용자가 글로벌 네트워크 환경하에서 쉽게 접근할 수 있도록 분산 객체 기술을 이용하여 3-Tier 클라이언트/서버 구조를 사용한다. *

참고문헌

[1] Bell, D., and Grimson, J., *Distributed Database Systems*, Addison-Wesley, 1992.

[2] Berson, A., *Client/Server Architecture*, McGraw-Hill, 1992.

[3] Bukhres, O., and Elmagarmid, A., *Object-Oriented Multidatabase Systems*, Prentice Hall, Englewood Cliffs, New Jersey, 1996.

[4] Date, C., *An Introduction to Database Systems*, Addison-Wesley, 1990

[5] Kamel, M., and Kamel, N., "Federated database management system: Requirements, issues and Solutions", *Computer communications*, Vol. 15, pp. 270 - 278, 1992.

* 본 연구는 1997년도 정보통신부 "CITIS/CALS 통합 DB 기술개발" 과제의 일부 연구내용임.

[6] Kidwell, R., and Richman, J., "Preliminary Integrated Weapon System Database (IWSDB) Implementation Strategy Paper", *Mantech International Corporation*, 1994.

[7] Ligon, T. S., *Client/Server Communications Services*, McGraw-Hill, 1997.

[8] Litwin, W., and Abdellatif, A., "Multidatabase Interoperability", *IEEE Computer*, Vol. 19, No. 12, pp. 10 - 18, 1986.

[9] Mowbray, T., and Zahavi, R., *The Essential CORBA*, Object Management Group, 1995.

[10] Object Management Group, *The Common Object Request Broker: Architecture and Specification 2.0*, July 1996.

[11] Orfali, R., and Harkey, Dan, *Client/Server Programming with JAVA and CORBA*, John Wiley and Sons, Inc., 1997.

[12] Phillips, J. F., *Reengineering Logistics*, CALS EXPO 96, USA.

[13] Sheth, A. P., and Larson, J. A., "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Database Systems," *ACM Computing Surveys*, Vol. 22, No. 3, pp. 183 -236, 1990.

[14] Smith, P., *Client/Server Computing*, McGraw-Hill, 1993.

[15] 정석찬,우훈식,백종명,주경준, "CITIS (Contractor Integrated Technical Information Services) 구현에 관한 고찰", *한국경영과학회/대한산업공학회 '97 춘계공동학술대회*, pp. 637-640, 1997.

[16] 우훈식,정석찬,백종명,주경준, "CALS 통합 데이터베이스 구현에 관한 연구", *한국경영과학회/대한산업공학회 '97 춘계공동학술대회*, pp. 641-644, 1997.