論文
5-9-22

기업간 통합 데이터 환경을 위한 데이터공유 지원 시스템의 설계 및 구현

Design and Implementation of Data Sharing Support System for Integrated Data Environment of Electronic Commerce for Business to Business

윤선희(Sun-Hee Yun)[1)]

요 약

최근 컴퓨팅 기술과 통신 기술의 급속한 발전으로 클라이언트/서버 컴퓨팅 환경에서 네트워크 컴퓨팅 시대를 지나 인터넷 컴퓨팅 시대가 도래하고 있다. 인터넷 사용이 보편화되어 감에 따라 기업의 정보 시스템이 인터넷 기반의 인트라넷/익스트라넷 시스템으로 구축되어 가고 있으며 미래의 비즈니스 환경은 기업의 이익 및 효율성을 최대화하기 위해 기업간 전자상거래가 보편화될 것으로 예상된다. <중략> 본 논문에서는 기업간 전자상거래의 통합 데이터 환경을 위하여 각 기업에 존재하는 기존의 시스템을 유지하면서 이질적인 데이터베이스들을 투명하게 접근할 수 있는 방법으로 웹 환경에서 Java/CORBA 기술, 관계형 및 객체지향형 데이터베이스와 파일 정보를 수용하기 위한 객체 질의 언어를 사용하는 데이터 공유 지원 시스템을 설계 및 구현한다.

Abstract

The area of business applications in the internet are extended enormously in result of fast development of computing and communication technologies, increase of internet use, and use of intranet/extranet in enterprise information system. 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 Electronic Commerce based on Business to Business(B2B). <중략>

This paper introduce the design and implementation of the data sharing support system that can be accessed data transparently by the users of participated enterprises in the integrated data environment supporting B2B Electronic Commerce. The system uses Java/CORBA technology in Web environment, relational and object-oriented database system, Object Query Language (OQL) to process the queries of the file information.

논문접수 : 2004. 10. 30.
심사완료 : 2004. 11. 15.

1) 정회원 : 숭의여자대학

본 논문은 숭의여자대학 교내학술연구비 지원에 의해 연구되었슴

1. 서 론

네트워크 컴퓨팅의 급속한 발전과 기업의 인터넷/익스트라넷의 일반적인 사용으로 미래의 비즈니스 환경은 기업의 이익 및 효율성을 최대화하기 위한 기업간 전자상거래가 일반화될 것으로 예상된다. 기업간의 정보 공유를 기반으로 하는 기업간 전자상거래를 운영하기 위해서는 통합 데이터 환경이 필요하며 이러한 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[9].

기업간 전자상거래의 데이터를 공유하기 위한 지원 환경의 기본 개념이 시스템을 새로 구축하는 것이 아니라 기존 시스템을 가능한 사용하는 것이기 때문에 기업간 전자상거래를 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 기업간 전자상거래를 위한 통합 데이터 환경을 지원할 수 있는 데이터 공유 지원시스템을 제안한다. 데이터 공유 지원 시스템은 데이터베이스 전위기(front-end)로써 데이터베이스 관리 시스템을 호출하는 응용 소프트웨어와 데이터베이스 관리기 사이에서 인터페이스 계층의 역할을 담당한다. 본 논문에서 제안된 데이터 공유 지원 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 데이터 공유 지원 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 데이터공유 지원 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

기업간 전자상거래의의 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[1,2].

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.

일본은 미국이 국방부의 주도하에 군무기 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 [4, 5].

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.

JCALS에서 구현한 GDMS는 하향식 접근 방식으로써 중앙 집중적 통제에 의한 수직적 분산 데이터베이스 시스템 형태의 통합 데이터베이스 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 객체 지향 데이터베이스 시스템에 존재할 수 있는 기업의 제품 데이터나 설계 도면 데이터의 직접적인 정보 공유가 불가능하다.

NCALS에서 구현한 수평적 분산 데이터베이스 시스템의 사용자 인터페이스는 HTML (HyperText Mark-up 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 객체 지향형 데이터베이스 시스템과의 연동을 고려하지 않고 있으며 동질 뿐만 아니라 이질의 데이터베이스 관리 시스템에서 제공하는 데이터들은 고려하지 않으며 단순히 파일들만을 공유되어지기 위한 데이터들로 고려한다.

3. 데이터공유 지원 시스템

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

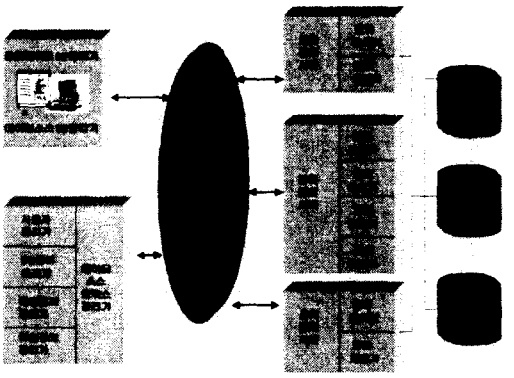
데이터 공유 지원 시스템은 이질적인 분산 데이터들의 논리적 통합에 의한 통합 데이터 환경을 지원하여야 하며 기업간 전자상거래의 플랫폼이 될 수 있는 인터넷 환경에서 사용자 질의 요청에 대해 전역적인 뷰(view)를 제공할 수 있도록 하며 일관성 있는 세션 관리가 되어야 한다. 각 기업의 자치성을 보장하며 기존에 사용 중인 시스템을 유지하기 위해 참여한 데이터베이스들의 동적인 스키마 통합에만 의존하지 않으며 기업간의 계약이나 정책에 따라 자동적으로 관리될 수 있는 영역에서 부분적 스키마 통합을 허용하여야 한다. 각 기업에 존재하는 이질의 하드웨어, 운영 체제, 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 객체 지향형 데이터베이스 시스템을 수용할 수 있어야 한다. 사용자와의 인터페이스를 위하여 다수의 사용자에 의해 동시에 데이터베이스를 처리할 수 있도록 설계되어 져야 하며 관계형 및 객체 지향형 데이터베이스의 질의를 처리할 수 있는 객체형 질의 언어가 요구된다.

데이터공유 지원 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 데이터공유 지원 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 소스(source) 들을 관리하기 위한 데이터 소스 관리 기능들로 구성된다.

본 논문에서 제안한 데이터공유 시스템의 구조는 다음과 같다(그림 1).



(그림 1) 데이터공유 지원 시스템 구조
(Fig. 1) Architecture of Database Sharing Support System

3.1 데이터 공유 지원 시스템의 구조

(1) 통합 데이터 접근 인터페이스

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

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

(2) 데이터 소스 인터페이스 관리기

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

데이터 소스 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 2는 전역 질의를 부질의로 분해하여 데이터 소스 인터페이스 관리기에 의해 지역 질의 서버에 전달하기 위한 인터페이스 및 각 지역 데이터베이스 시스템과의 매핑을 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 2) 전역 질의/지역 서브질의 I/F를 위한 ODL

(Fig. 2) ODL for Global Query/Local Subquery Interface(I/F)

(3) 데이터 소스 인덱스 관리기

데이터 사전/디렉토리의 역할을 담당하는 데이터 소스 인덱스 관리기는 클라이언트로부터 입력된 전역 질의가 전역 질의 서버에 의해 서브질의로 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. 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 3) 데이터 원시 정보 색인 IDL
(Fig. 3) 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 서브 질의로 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 OQL형식을 따른다. OQL은 관계형 데이터베이스의 SQL에 대응하는 문법으로 구성된다

전역 질의 서버에 접속하여 서비스를 받기 위해 필요한 IDL 파일은 다음과 같다(그림 4).

```
// 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 타
입
short     user_length;//user-defineddata타입
길이
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 4) 전역 질의 서버를 위한 IDL
(Fig. 4) Global Query Server IDL

(5) 지역 질의 서버

지역 질의 서버는 전역 질의 서버로부터 전달받은 전역 질의의 부질의문인 각 지역의 관

계형 DBMS의 데이터와 객체 지향형 DBMS 및 파일 정보를 처리하기 하기 위해 질의를 변환하여 처리하고 질의된 결과를 전역 질의 처리기에 전달하는 역할을 담당한다.

질의 처리를 한 후 질의 결과를 전역 질의 서버에 전달하며 각 지역의 관계형 데이터베이스의 데이터를 위한 질의 결과 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 타
입
string    user_type;  // user정의 data 타입
long      user_length; //해당data 타입  길이
};
typedef sequence<RMDField> RMetaData;

interface QueryResult {
```

```
RMetaData       getMetaData()
                  raises (LQSException);
RMDFieldgetRMDField(in long index)
                  raises (LQSException);
RMDFieldgetRMDFieldByName(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
sql_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 5) 지역 질의 서버를 위한 IDL
(Fig. 5) IDL for Local Query Server

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

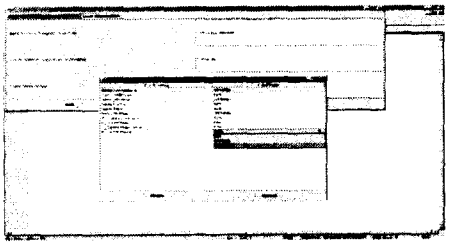
3.2 데이터공유 지원 시스템 수행 절차

제안된 데이터공유 지원 시스템의 수행절차는 다음과 같다.

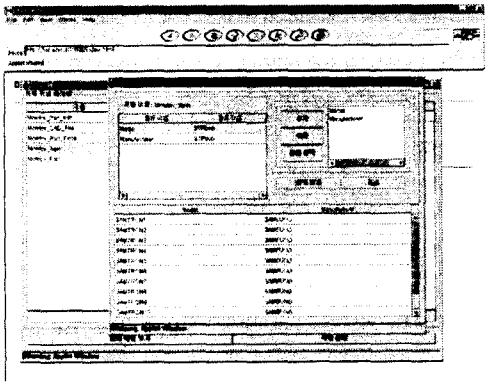
1) 사용자 인터페이스에 의해 사용자의 등급에 따른 엑세스 제어 절차에 따라 질의 입력이 허용된다.
2) 세션 관리기에 의해 세션 서버가 생성된다.
3) 세션 서버는 데이터 원시 정보 관리기에 지역질의 서버에 사상하기 위한 정보 및 파일 정보를 요청 한다.
4) 세션 서버는 사용자에게 전역 질의를 요청 받는다.
5) 세션 서버는 요청 받은 전역 질의를 전역 질 의 서버에 전달한다.
6) 전역 질의 서버는 전역 질의를 구문 해석하여 부질의를 생성한다.
7) 전역 질의 서버는 데이터베이스 인터페이스 관리기에 지역 질의 서버와의 접속을 요청한다.
8) 전역 질의 서버는 데이터 소스 인터페이스 관리기에 서브질의를 전달한다.
9) 데이터 소스 인터페이스 관리기는 해당 지역 질의 서버에 서브질의를 전달한다.
10) 지역 질의 서버는 전달 받은 서브질의를지역 질의 처리를 위해 변환하여 부질의를 처리

한다.
11) 지역 질의 서버는 서브질의 결과를 데이터 소스 인터페이스 관리기에 전달한다.
12) 데이터 소스 인터페이스 관리기는 부질의 결과를 전역 질의 서버에 전달한다.
13) 전역 질의 서버는 서브질의 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 해제하 고 세션을 종료한다.

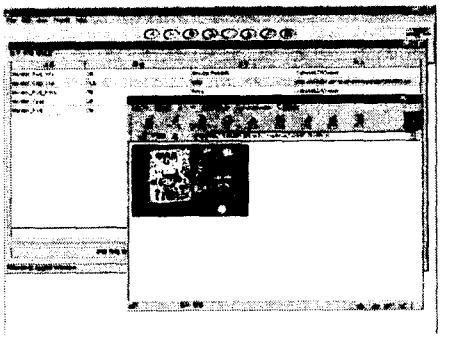
본 논문에서 구현된 예제 화면은 다음과 같다.



(그림 6) 데이터 소스 인덱스 관리기
(Fig. 6) Data Source Index Manager



(그림7) 선택된 객체 테이블에 대한 정보 화면
(Fig. 7) Selected Field Data Display Screen



(그림 8) 파일 타입 정보 화면
(Fig. 8) File Type Data Screen

3. 기대 효과

본 논문에서 제안된 데이터공유 지원 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 기업간 전자상거래의 기본 개념인 기존 시스템을 유지할 수 있도록 하였다.

제공되는 소스 데이터의 형태가 JCALS의 GDMS는 관계형 데이터베이스와 파일 형태의 데이터만을 고려하며 객체 지향형 데이터베이스 시스템에서 관리되는 도면 데이터나 객체 지향형 데이터에 대한 고려를 하지 않고 있으

며 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.

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 데이터공유 지원 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 데이터공유 지원 시스템은 기업간 전자상거래의 기본 개념인 기존 시스템을 사용하면서 각 기업간에 존재하는 지역 데이터베이스 응용 시스템의 자치성을 유지하며 통합 데이터 환경을 지원하기 위해 기존 시스템에서 제공하지 못한 기술적 문제점들을 해결하여 구현되었다.

4. 결론

본 논문에서는 "기업간 전자상거래의 실현"이라는 목적에 따라 기업간의 정보 공유를 위하여 기존에 사용 중인 시스템을 유지하면서 위치 투명성이 제공될 수 있는 통합 데이터 환경을 지원하는 데이터베이스 공유 지원 시스템의 프로토타입을 구현하였다.

본 논문에서 구현된 기업간 전자상거래의 핵심 기술인 통합 데이터 환경을 지원하는 데이터공유 지원 시스템은 기존의 시스템을 그대로 유지하여 기업의 자치성을 인정하며 이질 플랫폼상에서 물리적으로 분산된 데이터 소스들을 논리적으로 통합하여 사용자에게 위치 투명성을 제공하며 단일의 뷰로써 정보가 제공되어지도록 하여 "한번 생성된 데이터는 다수의 기업에 의해 공유되어 사용되어 져야 한다"는 기업간 전자상거래의 목적을 만족하는 시스템으로 제공될 수 있다.

참고문헌

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

[2] CSC, JCALS Developers Tool Kit, GDMS Implementation Guide(1997), Second Release

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

[4] Akira Nishiguchi, Koichiro Shida, Yuji Takada(1997), Steel Plant CALS Project, CALS Expo International 1997 Proceding

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

[6] 윤선희, 주경준, 정진욱(1998), "CALS 기반 분산 네트워크 하부구조에 관한 연구", 한국정보과학회 '98 춘계학술대회

[7] 윤선희, 우훈식, 문희철, 정승욱, 박상봉(1998), CORBA 기반 멀티데이터베이스 구현에 관한 연구", 한국 정보처리학회 '98 춘계학술대회

[8] 박치항, 이상구 역(1996), 분산 오브젝트 지향 기술 CORBA, 홍릉과학 출판사

[9] 김지운, 문강식, 김수웅, 이전영(1997), "Web과 멀티데이터베이스 시스템의 CORBA 기반 연동", pp. 323-328, HCI '97 학술대회 발

표 논문집
[10] 박현민, 이선미, 정효택(1997), "객체지향 메타 모델의 관계형 데이터베이스 스키마 변환", pp.343-346, 한국정보처리학회, '97 추계 학술 발표논문집
[11] 윤인중, 김홍남, 김창갑(1997), "웹환경하에서 자바를 이용한 효율적인 데이터베이스 접속에 관한 연구", pp 451-456, 한국정보처리학회, '97추계학술대회

윤 선 희
1983년 숭실대학교 전산학과 (학사)
1986년 웨인주립대학교 전산 학과(석사)
2000년 성균관대학교 정보공 학과 (박사)
1986년 ~ 1991년 CSDC, DUCOM, Inc.
시스템 분석가
1991년 ~ 2000.2 한국전자통신연구원
컴퓨터.소프트웨어기술연구소연구원
2000년 ~ 현재 숭의여자대학 인터넷정보과
교수
관심분야 : 분산처리응용, CALS/EC, HCI,
전자상거래 응용분야, 네트워크게임