인트라넷에서 가상데이터베이스를 이용한 데이터베이스 검색 시스템의 설계

이 동 욱† · 박 영 배††

요 약

현재 웹을 통하여 인터넷 데이터베이스를 검색하는 방법에는, 검색 엔진 기술을 이용한 방법과, 둘째 플러그인 기술이나 액티브엑스 기술을 이용한 검색 방법으로 나눌 수 있다. 검색 엔진을 이용하는 방법은 단순 문서와 같은 대량의 텍스트 데이터를 키워드와 같은 인덱스를 구축하고 이를 이용하여 검색하는 방법으로서, 문제점은 하나 이상의 데이터베이스를 동시에 검색하지 못하고 또 질의어와 같은 다양한 형태의 조건 검색을 할 수가 없으며, 사용자의 요구에 맞지 않는 데이터가 많이 전송된다는 즉, 정확성이 낮다는 세 가지 점을 들 수 있다.

플러그인 기술이나 액티브엑스 기술을 이용하는 방법은 웹 화면상에서 DBMS를 이용하여 클라이언트의 질의를 실행시켜 데이터베이스를 검색하는 방법으로서, 문제점은 동일한 데이터 모델의 경우라도 하나 이상의 DBMS를 동시에 기동시킬 수 없고 또 클라이언트 프로그램에서 미리 정의된 질의 이외의 다양한 종류의 질의를 할 수 없다는 두 가지 점을 들 수 있다.

본 논문에서는 이러한 문제점들을 해결하기 위해서 인터넷/인트라넷상에서 새로운 개념으로 작성한 가상데이터베이스를 이용하여 웹 화면을 통해서 다양한 종류의 질의를 직접 할 수 있는 데이터베이스 검색 시스템을 설계하는 데에 있다. 여기서, 가상데이터베이스는 동일한 관계 데이터 모델로서 하나 이상의 관계DBMS를 사용하는 것으로 가정한다.

Design for Database Retrieval System using Virtual Database in Intranet

Dong-wook Lee† · Youngbae Park††

ABSTRACT

Currently, there exists two different methods for database retrieval in the internet. First is to use the search engine and the second is to use the plug-in or ActiveX technology. If a search engine, which makes use of indices built from keywords of simple text data in order to do a search, is used when accessing a database, first it is not possible to access more than one database at a time, second it is also not possible to support various conditional retrievals as in using query language, and third the set of data received might include many unwanted data, in other words, precision rate might be relatively low.

† 정 회 원 : 한국정보통신기술사협회 부회장
†† 정 회 원 : 명지대학교 컴퓨터공학과 교수
논문접수 : 1998년 2월 10일, 심사완료 : 1998년 6월 9일

Plug in or ActiveX technology make use of Web browser to execute clients' query in order to do a database retrieval. Problems associated with this is that it is not possible to activate more than one DBMS simultaneously even if they are of the same data model, second it is not possible to execute a user query other than the ones that are previously defined by the client program.

In this paper, to resolve those aforementioned problems, we design and implement database retrieval system using a virtual database, which makes it possible to provide direct query interface through the conventional Web browser. We assume that the virtual database is designed and aggregated from more than one relational database using the same data model.

1. 서 론

웹(World Wide Web, WWW)[5]은 인터페이스의 편의성 때문에 많은 사람들이 쉽게 사용하고 있다. 이러한 웹의 급속한 보급으로 이제는 인터넷이라는 말이 곧 웹인 것처럼 인식되고 있다. 웹의 폭발적 인기 때문에 대량의 데이터를 저장하고 관리하는 데이터베이스와 웹을 연동 하려는 연구가 활발히 진행되고 있다.

현재 웹을 통하여 인터넷 데이터베이스를 검색하는 방법에는 첫째, 검색 엔진 기술[6]을 이용한 단순 문서 검색 방법과 둘째, 플러그인(plug-in) 기술이나 액티브엑스(ActiveX) 기술[4]을 이용한 데이터베이스 검색 방법으로 나눌 수 있다.

첫번째 방법은 기상정보, 인물정보, 주식정보, 가격정보 등 대량의 텍스트 데이터를 검색엔진을 이용하여 키워드와 같은 인덱스를 구축하고 이를 이용하여 검색하는 방법을 말한다. 따라서 이 방법에서의 데이터베이스란 텍스트데이터를 단순히 모아둔 데이터의 집합을 의미한다. 이 방법의 문제점은 하나 이상의 데이터베이스를 동시에 검색하지 못하고 또 질의어와 같은 다양한 형태의 조건 검색을 할 수가 없으며, 사용자의 요구에 맞지 않는 데이터가 많이 전송된다는 점 즉, 정확성(precision)[7]이 낮다는 세 가지 점을 들 수 있다.

두번째 방법은 최신 방법으로 네스케이프사의 플러그인 기술이나 MS사의 액티브엑스 기술을 이용하는 방법으로 웹 브라우져상에서 DBMS를 이용하여 클라이언트의 질의(Query)를 실행시켜 데이터베이스를 검색하는 방법이다. 이 방법의 문제점은 하나 이상의 DBMS를 동시에 가동시킬 수 없고 또 클라이언트 프로그램에서 미리 정의된 질의 이외의 다양한 종류의 질의를 할 수 없다는 문제점이 있다.

본 논문에서는 이러한 문제점들을 해결하기 위해서 인터넷/인트라넷상에서 새로운 개념으로 작성한 분야별 가상데이터베이스[1]를 이용하여 웹 화면으로 다양한 종류의 질의를 직접 할 수 있고, 하나 이상의 데이터베이스를 검색할 수 있는 가상데이터베이스 검색시스템을 설계하는 데에 있다. 여기서, 가상데이터베이스는 동일한 관계 데이터 모델로서 하나 이상의 관계DBMS를 사용하는 것으로 가정한다.

2. 관련 연구

웹과 데이터베이스의 연동은 크게 두 가지 방향으로 진행되고 있다. 첫번째는 인터넷이라는 거대한 네트웍에서 보다 쉽게 원하는 문서를 찾기 위한 문서 검색의 방법과, 두번째는 데이터베이스를 검색하기 위한 클라이언트 서버 시스템에서의 클라이언트 프로그램을 웹 브라우저 상에서 기동하는 방법이 있다.

첫 번째 방법은, 주로 하나 혹은 그 이상의 키워드를 사용하여 질의를 하며 사용자는 자신이 어떤 시스템의 어떤 데이터베이스(단순 문서들의 텍스트 데이터)를 사용하는지 알 수 없도록 되어 있다. 이러한 방법은 주로 CGI(Common Gateway Interface) 프로그램[9], 확장 API(Application Program Interface)[3], 스크립트(Script)[3]등을 이용하여 구현한다. 이러한 방법은 서비스 개발자가 미리 질의의 패턴을 만들어 놓고, 사용자는 빈칸에 키워드를 채우는 식으로 질의하는 방법이다. 이 방법의 대표적인 서비스 형태로는 도서관의 도서검색 시스템으로써, 책제목을 질의하는 질의 패턴

을 미리 만들어 놓고 사용자가 원하는 책의 제목을 입력하면 이를 미리 만들어 놓은 질의 패턴의 SELECT 절에 삽입하여 질의를 수행하는 방법이다.

CGI 프로그램을 이용하는 방법은 서버쪽에 데이터베이스를 검색하기 위한 CGI프로그램을 준비하여 놓고 클라이언트의 요청이 있을 때마다 이 프로그램을 기동시키는 방법이다. 이 방법은 클라이언트의 요청 수만큼 데이터베이스로의 연결을 위한 프로세스가 생성되므로 서버의 성능이 급격히 저하된다는 문제점이 있다. 확장 API를 사용하는 방법은 데이터베이스 제공자가 제공하는 데이터베이스 접속 API를 이용하여 데이터베이스에 직접 연결하는 어플리케이션을 작성하는 방법이다. 이 방법은 어플리케이션이 각각의 클라이언트 쪽에 존재하므로 서버의 부담이 줄어드는 장점은 있다. 스크립트 및 외부 프로그램을 이용하는 방법은 자바(JAVA)와 같은 스크립트 언어로 작성하거나 혹은 데이터베이스 접속 응용프로그램을 사용하여 데이터베이스에 접속하는 방법이다. 이 방법은 다양한 종류의 응용프로그램 개발 도구를 그대로 사용할 수 있다.

이러한 방법들의 공통적인 문제점으로는, 하나 이상의 데이터베이스를 동시에 검색하지 못하고 또 질의어와 같은 다양한 형태의 조건 검색을 할 수가 없으며, 사용자의 요구에 맞지 않는 데이터가 많이 전송된다는 점 즉, 정확성(precision)[7]이 낮다는 세 가지 점을 들 수 있다.

두 번째 방법은, COM(Component Object Model)[4]이나 DCOM(Distributed Component Object Model)[4] 개념에 기반한 넷스케이프(Netscape)사의 플러그인이나 마이크로소프트사의 액티브엑스[4] 기술을 이용하여 데이터베이스 검색을 위한 프로그램을 웹 브라우저 상에서 실행시키는 방법으로 사용자가 직접 필요한 질의문을 만들어서 질의를 하는 것이 가능하다.

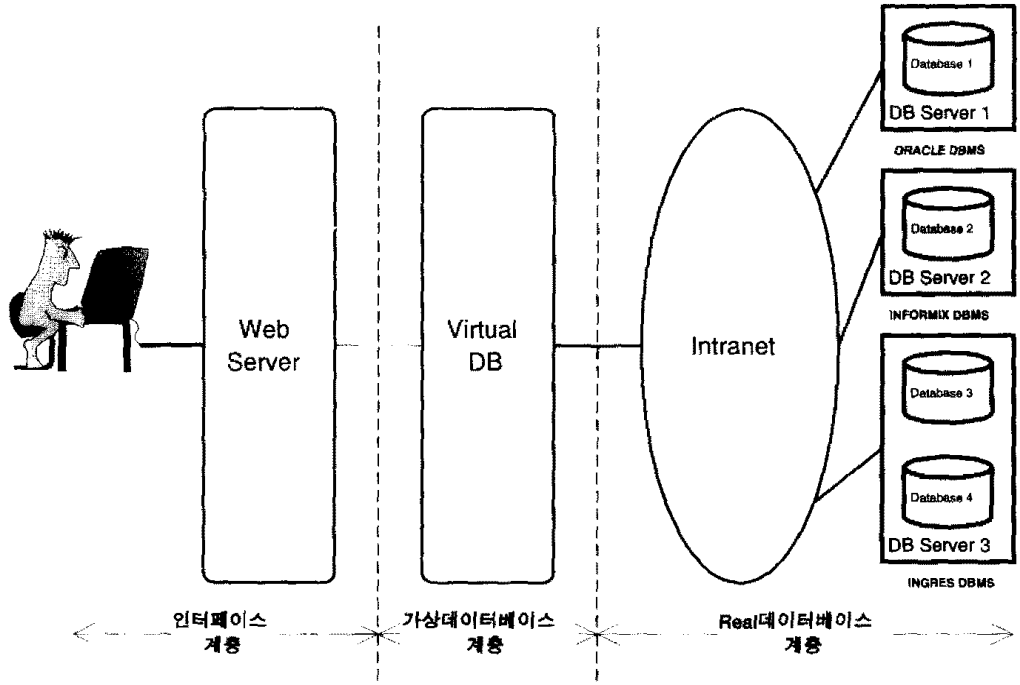
COM, DCOM으로 불리는 분산 객체 개념은 데이터베이스의 접근을 모두 클라이언트 쪽의 객체를 이용하여 수행하는 것이다. 분산 객체는 데이터를 검색하고 조작하는데 필요한 방법을 보유하고, 각종 응용프로그램들은 이 객체를 지속적으로 재사용 할 수 있다. COM 아키텍처를 사용할 경우에는 클라이언트 인터페이스를 통해서 COM 객체를 사용하며, 이 인터페이스는 프록시(Proxy)를 통해서 액세스된다. COM, DCOM을 이용한 플러그인이나 액티브엑스 기술은 응용프로그램간의 데이터 교환을 위한 기술을 인터넷에 확장한 것으로, 사용자가 작성한 질의문을 직접 데이터베이스 서버에 전달하는 것을 가능하게 한다. 예를 들면, 비주얼 베이직으로 데이터베이스에 접속하는 SQL 터미널 에뮬레이터 프로그램을 작성하였을 경우 액티브엑스 기술을 이용하면 이 프로그램을 웹 브라우저 속에서 직접 실행시킬 수 있다.

이러한 COM, DCOM에 기반한 액티브엑스 기술을 이용할 때의 장점은 다음과 같다. 첫째, 데이터베이스의 스키마가 변해도 객체가 가지고 있는 데이터베이스 조작방법만 바꾸면 웹을 통한 검색 방법에는 아무 변화가 없다. 이 점이 스크립트나 외부 프로그램을 사용할 때와의 차이점이다. 둘째, 데이터베이스 및 서비스 개발자가 클라이언트들에게 제공되는 데이터베이스 접속 방법을 통합적으로 관리할 수 있다. 셋째, C와 같은 언어를 사용할 경우 직접 데이터베이스에 질의를 하는 프로그램의 개발도 가능하다. 단점은 다음의 네 가지를 들 수 있다. 첫째, 데이터베이스를 검색하는 사용자와 데이터베이스 사이에 COM 객체가 중간 과정으로 개입되어 데이터베이스 검색의 효율을 떨어뜨리는 점과 둘째, 클라이언트에서 COM 객체와의 메시지 교환을 위하여 많은 API호출을 하기 때문에 네트웍 트래픽이 증가한다. 셋째, 프로그램 개발과정에서 객체의 메시지를 처리하기 위한 서버 프로그램이 작성되어야 하기 때문에 초기 개발에 많은 어려움이 따르고 넷째, 하나 이상의 DBMS를 동시에 검색할 수 없다는 점이다.

3. 가상데이터베이스 검색시스템의 3 계층

가상데이터베이스 검색시스템은 (그림 1)에서 보는 바와 같이 논리적으로 3 계층으로 나누며, 중간 계층인 가상데이터베이스(Virtual Database : VDB로 약칭)는 실데이터베이스(Real Database : RDB로 약칭)와 웹서버(Web Server)의 중간에 위치한다.

첫 번째 계층인 RDB 계층은 (그림 1)에서 보는 바와 같이 여러 사이트(site)에 서로 다른 데이터베이스 서버들이, 예를 들어 ORACLE, INFORMIX,



(그림 1) 가상 데이터베이스 검색시스템의 3 계층
(Fig. 1) Three Layers of VDB Retrieval System

INGRESS DBMS등이 존재한다. 두 번째 계층인 VDB 계층은 여러 사이트(site)에 흩어져 있는 서로 다른 RDB들로부터 스키마 정보들을 추출하여 저장해 두었다가, 사용자가 웹을 통해 질의를 하면 RDB에서 데이타를 검색하고 그 처리 결과를 인터페이스 계층에 보내 준다. 세 번째 계층인 인터페이스 계층은 웹 사용자가 VDB의 스키마 정보를 웹 화면으로 질의할 수 있도록 하고, 또 VDB 계층에서 보내온 처리 결과를 웹 화면으로 보여 준다.

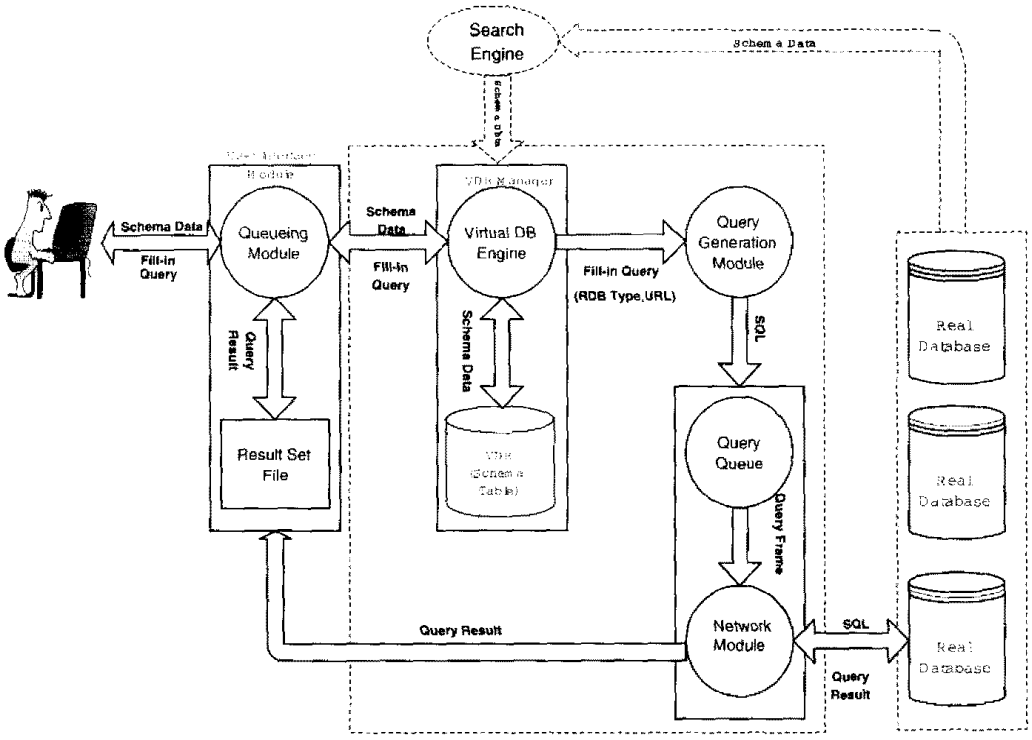
4. VDB 시스템의 구성

VDB 시스템은 (그림 2)와 같은 VDB 검색시스템에서 점선 부분을 말하며 다음과 같이 세 모듈로 구성되어 있다. 첫째, RDB의 스키마 정보를 모아서 스키마 테이블(Schema Table)에 저장하고 관리하는 VDB 관리자(Manager) 둘째, 웹 화면을 통한 필-인(Fill-In, 채워넣기) 방식의 질의어를 목표 DBMS에 맞는 질의문으로 변환해주는 질의어 생성 모듈(Query Generation Module) 셋째, 물리적으로 떨어진 원격지의 RDB에 접속하여 질의문을 전송하고 그 처리 결과를 받는 네트웍 모듈(Network Module)로 구성되어 있다.

데이터베이스 시스템에서는 실데이터가 보조기억장치에 저장되나, VDB는 RDB들의 스키마 정보만을 갖고 있는 특수한 형태의 데이터베이스이다. 사용자는 오직 웹을 통해서만 VDB의 스키마 데이터(Schema Data)를 볼 수 있으며, 큐잉 모듈(Queueing Module)을 통해서 필-인 질의(Fill-In Query)를 할 수 있다. 큐잉 모듈은 각각의 사용자 접속에 대하여 아이디(ID)를 발급하고 필-인 질의를 VDB 관리자에게 전달한다. VDB 관리자는 전달받은 필-인 질의와 RDB의 형식(Type)과 URL(Uniform Resource Location)를 질의어 생성 모듈에 전달하고, 질의어 생성 모듈은 이를 SQL 질의문으로 변환하여 네트웍모듈에 전송한다. 네트웍 모듈은 RDB의 DBMS를 기동시켜서 질의를 처리하고 그 결과를 반환받아 결과집합파일(Result Set File)에 전달한다. 큐잉 모듈은 이 결과집합파일을 참조하여 사용자에게 질의 결과를 웹 브라우져를 통하여 보여 준다. VDB 시스템은 실제의 DBMS 엔진을 가지고 있지 않다.

4.1 VDB 관리자

VDB 관리자는 (그림 2)에서 보는 바와 같이 VDB 엔진(Virtual DB Engine)과 스키마 데이터를 저장한



(그림 2) 가상데이터베이스 검색 시스템의 구성
(Fig. 2) Organization of VDB Retrieval System

VDB로 구성되어 있다. VDB 엔진은 다음과 같은 두 가지 기능을 갖고 있다. 첫째, 검색 엔진(Search Engine)이 각 사이트의 RDB를 검색하여 얻은 스키마 정보(스키마 이름, 스키마의 위치정보-URL, 데이터 타입)를 받아서 VDB(스키마 테이블)에 저장하고 관리한다. 둘째, 사용자가 웹을 통하여 VDB에 접속을 시도하면 사용자 인터페이스 모듈이 만든 웹 화면을 통하여 사용자에게 스키마를 보여 주기 위해 필요한 스키마 정보를 큐잉 모듈에게 제공한다. 또 큐잉 모듈로부터 필-인 질의(스키마, 질의 조건)를 받아서 질의어 생성 모듈에게 전달한다.

VDB의 스키마 테이블(Schema Table)은 〈표 1〉에서 보는바와 같이 테이블 정보와 열 정보의 두 테이블로 구분되어 저장된다. "테이블 정보" 테이블에는 각각의 테이블의 URL, 데이터베이스 이름, 테이블 이름, 데이터베이스 타입의 정보가 저장되고 "열 정보" 테이블에는 각각의 열들에 대한 정보가 저장된다.

VDB 관리자는 스키마 테이블을 맴버 데이터로 갖

〈표 1〉 스키마 정보를 표시하기 위한 테이블 구조
〈Table 1〉 Table Structures for Schema Information

테이블 정보				
테이블 번호	URL	데이터베이스이름	테이블 이름	데이터베이스 타입

열 정보				
테이블 번호	열 번호	이름	크기	속성

는 객체로 정의하며, VDB 관리자가 제공하는 퍼블릭 메소드(Public Method)는 다음과 같다.

```
class VDB_Manager {
    private :
        FILE SchemaTable;      // 스키마 정보
        가 저장된 데이터베이스 파일
                    .
                    .
                    .
    public :
      CSchema* GetSchema();        // 사용자
      인터페이스 모듈에 의하여 스키마정보를 읽는
      메소드
      CSchema  GetSchema(CString  URL);
       //주어진 URL을 가지고 스키마를 읽어 오는
      메소드
      CString GetURL(CSchema schema);
      // 주어진 스키마에 해당하는 URL을 찾는 메
      소드
      int GetDBType(CString URL); // 주어
      진 URL을 가지고 데이터베이스 형식을 읽어
      오는 메소드
                    .
                    .
                    .
}
```

VDB_Manager 클래스의 GetSchema() 메소드는 사용자 인터페이스모듈에 의하여 호출되며 VDB가 가지고 있는 스키마 정보들을 CSchema 형태의 포인터로 반환한다. GetSchema(CString URL) 메소드는 URL을 파라메터로 받아서 해당하는 URL에 위치한 데이터베이스의 스키마 정보를 반환하고 반대로 GetURL() 메소드는 스키마를 파라메터로 받아 스키마의 URL을 반환한다. GetDBType() 메소드는 URL을 파라메터로 받아서 해당하는 데이터베이스를 관리하고 있는 DBMS의 타입을 반환한다.

4.2 질의어 생성 모듈

질의어 생성 모듈은 (그림 2)에서 보는 바와 같이 VDB 관리자로부터 필드인 질의와 RDB 데이터(스키마, 질의 조건, URL, 데이터 타입)를 받아서 네트웍 모듈이 RDB를 검색할 수 있도록 각각의 DBMS에 맞는 질의문으로 변환하여 네트웍 모듈에게 제공한다.

실제로 데이터가 저장되어 있는 각각의 RDB에 적합한 질의문을 생성하는 방법은, 첫째 각각의 DBMS가 모두 지원하는 표준 SQL을 이용하는 방법과, 둘째 스키마 테이블에 저장되어 있는 RDB 타입 정보를 이용하여 목표 DBMS에 적합한 질의문을 생성하는 전용 질의어 생성기를 사용하는 방법이 있다. 본 논문에서는 첫째 방법을 사용한다.

4.3 네트웍 모듈

네트웍 모듈은 RDB의 URL을 이용하여 RDB에 접속하고 질의어 생성 모듈이 변환한 질의문을 이용하여 RDB의 목표 DBMS를 기동(invoke)시켜서 DBMS로 하여금 RDB를 검색하게 하고 그 처리 결과를 받아서 사용자 인터페이스 모듈의 결과집합파일에 전달한다. 이렇게 설정된 VDB와 RDB의 접속은 질의어의 처리 결과가 네트웍 모듈에게 되돌아올 때까지 유지되고 결과가 돌아오면 접속이 종료된다. 또 네트웍 모듈은 하나 이상의 네트웍 접속 요청을 동시에 받을 수 있기 때문에, 하나의 질의에 대한 처리 결과를 받기 전에 다른 질의문을 수행을 할 수 있어야 하므로 이를 위해 큐를 사용하여 질의어 생성기로부터 도착한 질의문을 큐에 저장하고 한번에 하나씩 질의문을 처리하게 한다. 네트웍 모듈에서 사용하는 큐 객체의 구조는 다음과 같다.

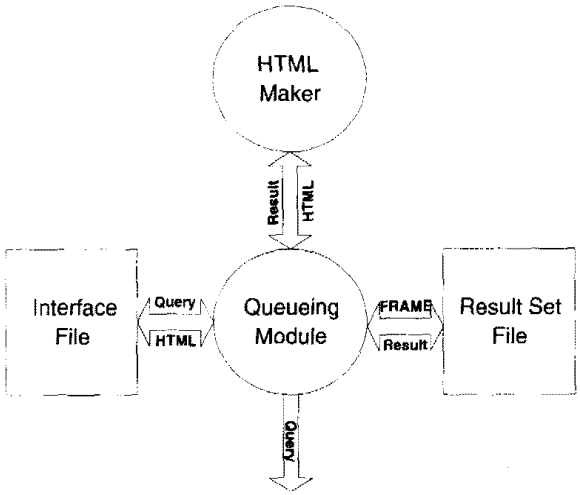
```
CLASS QueryFrame : CList { ..... }
CLASS QueryQueue {
{
        private :
        QueryFrameList mQueue;
                    .
                    .
                    .
        public :
        int              AddQuery(QueryFrame
&mFrame);
        QueryFrame * DeleteQuery();
}
```

QueryQueue 클래스는 각각의 사용자들로부터 요구된 질의들을 순서대로 저장하는 원형 큐(Circular Queue)이다. 두 개의 퍼블릭 메소드 AddQuery()와

DeleteQuery()는 사용자 인터페이스 모듈에 의하여 호출되며, DeleteQuery() 메소드는 원형 큐의 머리(head)에 있는 질의 프레임 객체를 꺼내오고, AddQuery() 메소드는 새로운 질의가 생성됐거나 혹은 DeleteQuery()를 이용하여 꺼냈던 질의 프레임에 해당하는 결과가 아직 생성되지 않았을 경우 그 질의 프레임을 큐에 다시 넣을 때 사용한다.

5. 사용자 인터페이스 모듈

사용자 인터페이스 모듈은 (그림 2)에서 보는 바와 같이 사용자와 VDB간의 인터페이스를 담당하기 위하여 다음의 세 가지 기능을 수행한다.



(그림 3) 사용자 인터페이스 모듈의 구성
(Fig. 3) Organization of User Interface Module

첫째, 사용자가 VDB 검색시스템의 웹 화면에 연결하면 VDB 관리자로부터 스키마 정보를 받아서 웹 화면에 표시해 준다. 둘째, 사용자가 웹으로 표시된 스키마 정보를 보면서 필-인 방식으로 질의를 하면 그 필-인 질의를 VDB 관리자에게 전달한다. 셋째, 질의의 결과를 사용자가 볼 수 있는 형태로 가공하여 웹 화면에 표시해 준다. 이때 스키마 정보와 질의어의 처리 결과를 웹상에 표시하기 위해서는 반드시 HTML Maker를 이용하여 HTML 문서로 변환해야 한다. 또 사용자 인터페이스 모듈은 다수의 사용자 또는 다수의 요구 사항을 동시에 처리하기 위해서 큐잉 모듈과 결과집합파일 모듈을 사용한다.

특히, 사용자 인터페이스 모듈은 새로운 질의가 생성될 때마다 그 질의어의 처리 결과로 제공할 결과 파일의 템플릿(Template)을 동적으로 생성한다. 이렇게 생성된 결과 파일 템플릿은 뒤에 HTML Maker에서 만든 HTML 문서로 그 내용이 채워질 때까지 사용자와의 연결이 끊어지지 않도록 하는 역할을 한다.

하나의 클라이언트가 사용자 인터페이스 모듈을 통하여 질의를 하면, 큐잉 모듈은 각각의 질의마다 유니크(Unique)한 아이디(ID) 번호를 발급하고 그 질의를 생성한 클라이언트의 아이피 주소(IP Address), 소켓 번호(Socket Number)와 함께 하나의 프레임을 만들어 큐에 저장한다. 이 때 사용하는 프레임 구조는 다음과 같다. 이렇게 저장된 아이피 주소와 소켓 번호는 질의어의 처리 결과를 클라이언트에게 전송할 때 사용된다.

```
struct QueryFrame {
    unsigned int ID; // 사용자 인터페이스 모듈
    이 만든 유니크한 아이디
    unsigned long IP_Address; // 질의를 한 클
    라이언트의 주소
    int SocketNum; // 질의를 한 클라이언트의
    소켓 번호
    }
```

위의 아이디(ID) 번호를 유일하게 발급하기 위한 방법으로 원형 큐를 이용한다. 원형 큐는 초기화 과정에서 0부터 65535까지의 숫자를 순차적으로 채워 넣는다. 질의가 생성된 경우에는 원형 큐의 헤드(head)에서 하나의 아이디(ID)를 꺼내서 발급하고 결과가 생성되면 사용했던 아이디를 회수하여 원형 큐의 꼬리(tail)에 저장한다. 큐잉 모듈에 프레임 정보가 삽입된 후 클라이언트가 생성한 질의는 VDB 관리자에게 전달된다.

결과집합파일은 네트웍 모듈로부터 전달받은 질의어의 처리 결과를 저장하며 결과집합파일의 구조는 헤더(Header) 부분과 내용(Content) 부분으로 구성된다. 헤더 부분에는 〈표 2〉에서 보는 바와 같이 결과집합파일에 저장된 각 질의어들의 처리 결과에 대한 정보가 기록되고, 사용자 인터페이스 모듈은 질의어의 처리 결

과가 저장되었는지를 검색할 때 검색 속도를 향상시키기 위하여 이 정보를 이용한다. 헤더 부분의 Result Array는 Result라는 구조체의 배열이다. Result 구조체의 멤버는 각각의 질의에 발급된 아이디, 결과집합파일의 내용 부분에서 아이디에 해당하는 결과가 시작되는 옵셋, 처리 결과의 크기로 구성되어 있다. 내용 부분은 질의 처리결과를 텍스트 형태로 저장하고 있다.

〈표 2〉 결과 집합 파일의 헤더 구조
〈Table 2〉 Header Format in Result Set File

이 름	크 기	내 용
Num	2byte	결과 집합 파일에 저장된 결과의 수 헤더에 저장된 Result의 갯수
Size	4byte	헤더의 크기를 표시
Result Array	10byte * Num	각각의 결과들에 대한 정보가 들어있는 구조체의 배열

사용자 인터페이스 모듈은 결과집합파일에 내용이 추가될 때마다 그 파일의 헤더를 검사하여 현재 큐의 헤드에 위치한 프레임의 아이디(ID)와 같은 아이디(ID)를 갖는 Result 구조체가 결과집합파일의 헤더 부분에 있는지의 여부를 검사한다. 만약, 일치하는 아이디(ID)를 갖는 Result 구조체가 결과집합파일의 헤더에 있다면 해당하는 Result 구조체를 결과집합파일의 헤더로부터 읽고 그 정보를 이용하여 질의어의 처리 결과를 결과집합파일의 내용 부분에서 읽어 온다. 만약 없다면, 프레임은 다시 큐의 꼬리에 삽입된다.

사용자 인터페이스 모듈은 결과집합파일에서 읽어온 처리 결과를 HTML Maker를 이용하여 HTML 문서로 만들고, 프레임에 저장된 IP 주소와 소켓 번호를 이용하여 어떤 클라이언트에게 제공할 것인지 결정한다. 클라이언트로부터 질의가 생성된 경우와 결과집합파일을 검사하는 과정의 알고리즘은 다음과 같다.

```
EVENT : QueryGenerated
MESSAGE : MakeFrame
METHOD :
   newFrame(MakeUniqueID(),Client.IP,
   Client. Socket);
```

새로운 질의가 생성되었을 때 Frame 생성자가 수행된다. Frame 생성자는 질의 ID와 질의를 한 클라이언트의 IP 그리고 클라이언트의 소켓 번호를 파라메터로 갖는다.

```
EVENT : NewResultStored
MESSAGE : ResultFileUpdated
METHOD :
HeadFrame = Queue.delete();
startflag = HeadFrame.GetID();
do {
 if  (ResultSetFile.QueryDone(HeadFrame.
 QueryID)) {
   ResulrSetFile.GetResultInfo(ResultInfo);
   ResultSetFile.GetQueryResult(ResultInfo,
   Buf);
   HTMLMaker.MakeResultFile(Buf);
   }
 else Queue.Add(HeadFrame);
 HeadFrame = Queue.delete();
  } while ( startflag != HeadFrame.GetID())
}
```

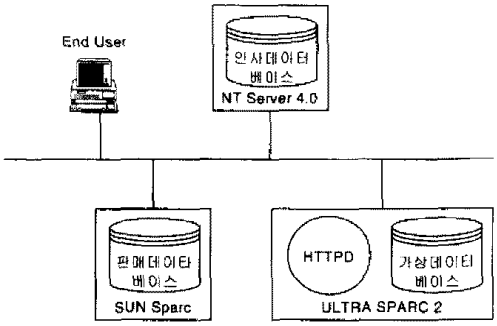
결과집합파일이 갱신되었다는 메시지가 발생하면 큐 객체에 있는 질의 프레임들을 하나씩 꺼내어(Queue. delete() 메소드) 생성된 결과를 요구한 질의인가를 확인한다(QueryDone() 메소드). 만약 해당하는 결과를 요구한 질의가 맞다면, 결과를 HTML 문서의 모양으로 바꾸어 전송하고(if 절) 그렇지 않으면, 큐에 다시 넣은 후(else 문) 큐를 완전히 다 검사할동안 위의 작업을 반복한다(do ~ while 루프).

6. 시스템 구현 사례

시스템 구현은 스포츠 용품을 판매하는 A회사의 B지점 영업부에서 사용하는 판매 데이터베이스와 A회사의 본사 인사부에서 사용하는 사원인사 데이터베이스를 동시에 접근하기 위한 시스템을 VDB를 이용하여 실험중에 있다. 구현에 있어서의 제약 조건으로는 각각의 데이터베이스가 모두 관계형 데이터 모델을 사용하고 있는 경우를 가정하였다.

6.1 시스템 구현 환경

본 논문에서 사용한 VDB 검색시스템의 구성은 (그림 4)와 같고 판매 데이터베이스는 인포믹스사의 Universal Server(SUN Sparc)에, 인사 데이터베이스는 마이크로소프트사의 SQL Server(NT Server)에, 이들을 이용한 VDB는 인포믹스사의 Universal Server (ULTRA Sparc)에 구축하였다. 판매 데이터베이스 및 인사 데이터베이스의 스키마 구조는 〈표 3〉과 〈표 4〉와 같다. 판매 데이터베이스에는 고객, 주문, 제품목록, 주문제품 테이블들이 있고, 인사 데이터베이스에는 사원, 조직, 인사, 가족관계 테이블들이 있다.



(그림 4) 시스템 구현 환경
(Fig. 4) Structure of VDB System

〈표 3〉 판매데이터베이스의 스키마
〈Table 3〉 Schema of Sales DB

고객					
고객번호	고객이름	회사명	주소	우편번호	전화번호

주문			
주문번호	고객번호	주문일자	지불일자

제품목록		
제품번호	제품명	제품설명

주문제품			
제품번호	주문번호	수량	가격

〈표 4〉 인사 데이터베이스의 스키마
〈Table 4〉 Schema of Personality DB

사원					
사원번호	이름	주민등록번호	주소	전화번호	부서명

조직	
부서번호	부서명

인사	
사원번호	인사내용

가족관계			
사원번호	관계	이름	주민등록번호

〈표 5〉 가상 데이터베이스의 테이블 정보
〈Table 5〉 Table Information of VDB

테이블 정보				
테이블번호	URL	데이터베이스 이름	테이블 이름	데이터베이스 타입
1	202.30.101.104	판매데이터베이스	고객	Universal Server
2	202.30.101.104	판매데이터베이스	주문	Universal Server
3	202.30.101.104	판매데이터베이스	제품목록	Universal Server
4	202.30.101.104	판매데이터베이스	주문제품	Universal Server
5	202.30.101.193	인사데이터베이스	사원	SQL Server
6	202.30.101.193	인사데이터베이스	조직	SQL Server
7	202.30.101.193	인사데이터베이스	인사	SQL Server
8	202.30.101.193	인사데이터베이스	가족관계	SQL Server

6.2 VDB의 테이블 구조

ULTRA Sparc에 있는 VDB는 위의 두 데이터베이스의 스키마 정보를 저장하고 있다. VDB의 테이블 정보는 〈표 5〉와 같고, 속성 정보는 〈표 6〉과 같다.

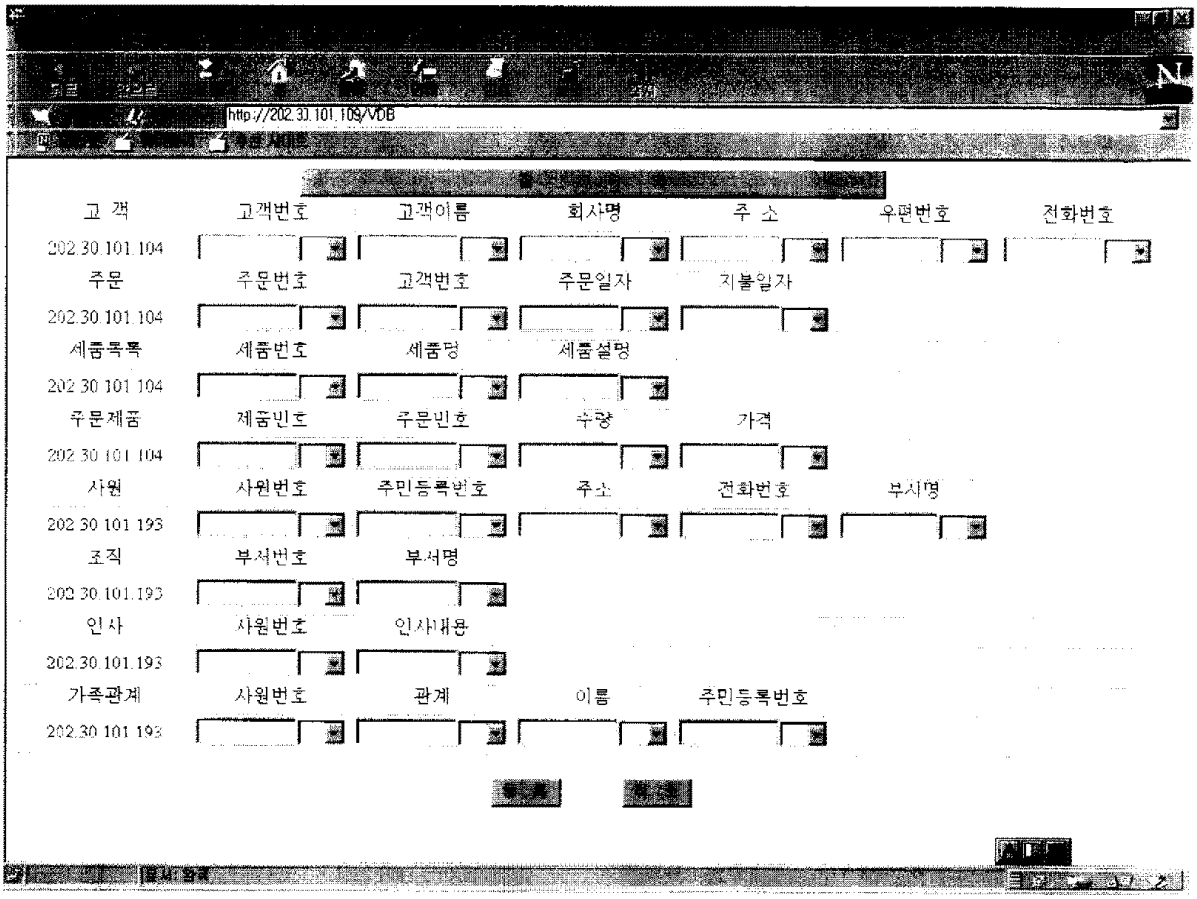
사용자가 웹을 통하여 VDB에 접속을 요청하면, 웹 화면에 〈표 5〉와 〈표 6〉의 스키마 정보를 나타내는 데에는 다음과 같은 방법이 있다. 첫째는 "테이블 정보" 테이블로부터 각각의 테이블번호에 해당하는 데이터베이스 이름과 테이블 이름을 찾아낸 후 다시 "속성 정보" 테이블에서 각각의 테이블에 해당하는 속성들을 찾아내어 표시하는 방법이다. 이 방법은 질의문을 반복적으로 VDB에 전달할 수 있는 방법을 필요로 한다. 두 번째 방법은 "테이블 정보" 테이블과 "속성 정보" 테이블을 조인하는 방법으로 테이블 번호를 키로 하여 두 테이블을 조인하면 된다. 본 논문에서는 조인 연산을 피하기 위하여 첫 번째 방법을 사용하고 있다. 질의문을 VDB에 반복적으로 전달하기 위한 방법은 인포믹스 유니버설 서버에서 제공하는 "for" 문을 이용하여 저장 프로시져로 구현하고 있다.

6.3 웹을 이용한 질의 예

VDB 관리자로부터 사용자 인터페이스 모듈을 통하여 웹 화면에 표시되는 스키마 정보는 (그림 5)와 같은 형태로 표현된다. (그림 5)에 표시된 그리드 구조에서 두 개의 행이 쌍을 이루어 하나의 테이블을 표시하고 있다. 홀수 번째 행은 각각의 테이블들을 표시하고 있고, 짝수 번째 행은 질의를 하기 위한 입력 도구로 되어 있다. 테이블을 표시하는 홀수 번째 행의 첫 번째 속성은 테이블 이름이고 두 번째 속성부터는 각각의 테이블의 속성 이름을 표시하고 있다. 짝수 번째 행의 첫

〈표 6〉 가상 데이터베이스의 속성 정보
〈Table 6〉 Attributes Information of VDB

속성 정보				
테이블 번호	열번호	이름	크기	속성
1	1	고객번호	4	int
1	2	고객이름	8	char
1	3	회사명	20	char
1	4	주소	40	char
1	5	우편번호	6	char
1	6	전화번호	12	char
2	1	주문번호	4	int
2	2	고객번호	4	int
2	3	주문일자	4	date
2	4	지불일자	4	date
3	1	제품번호	4	int
3	2	제품명	20	char
3	3	제품설명	100	char
4	1	제품번호	4	int
4	2	주문번호	4	int
4	3	수량	4	int
4	4	가격	4	money
5	1	사원번호	4	int
5	2	이름	8	char
5	3	주민등록번호	13	char
5	4	주소	40	char
5	5	전화번호	12	char
5	6	부서명	20	char
6	1	부서번호	4	int
6	2	부서명	20	char
7	1	사원번호	4	int
7	2	인사내용	200	char
8	1	사원번호	4	int
8	2	관계	4	int
8	3	이름	8	char
8	4	주민등록번호	13	char



(그림 5) 사용자 인터페이스의 웹 화면
(Fig. 5) Web Page of User Interface

번째 속성은 각각의 테이블이 위치하는 서버의 URL과 데이터베이스 이름을 표시한다.

테이블의 URL과 데이터베이스 이름을 동시에 표시하기 위하여 본 논문에서는 e-mail 주소를 표시하는 방법을 이용하였다. 본 논문에서 NT Server에 위치한 인사 데이터베이스는 인사@ns.ce.myongji.ac.kr과 같은 방법으로 표시한다.

VDB에 질의를 하기 위해서 사용자는 먼저 웹 화면에서 원하는 속성 이름을 찾아야 한다. 사용자 인터페이스는 그리드의 형태로 각각의 테이블 명과 속성 이름을 표시하고 있으므로 사용자는 먼저 필요한 속성 이름을 고르고 자신이 선택한 속성이 원하는 테이블에 있는지를 확인한다. 필요한 속성이 선택되면 사용자는 그리드에서 자신이 선택한 속성에 해당하는 셀의 바로 아래 위치한 에디트 박스(edit box)에 검색 조건을 입력한다. 또 질의의 조건이 입력된 속성 정보 이외의 임의의 속성 정보에 대해서 "*"연산자를 적용시키면 해당하는 테이블에서 조건에 만족하는 투플의 모든 속성을 표시한다. 임의의 속성만을 표시할 때는 원하는 속성에만 "?"연산자를 적용시킨다. 이러한 방법을 필-인 방법이라고 한다.

(질의 예)

판매데이터베이스에 있는 고객 테이블에서 권영일이라는 이름의 고객의 전화번호를 알고 싶다면 먼저 판매@ape.ce.myongji.ac.kr이라는 위치정보와 고객이라는 이름을 갖고 있는 테이블을 찾는다. 다음으로 이름이 "권영일"인 고객을 찾으려는 것이므로 "이름"이라는 속성 정보 아래의 에디트 박스에 "= 권영일"이라고 입력한다. 마지막으로 찾으려는 정보는 전화번호이므로 "전화번호"라는 속성 정보 아래의 에디트 박스에 "?"를 입력하고 "질의"버튼을 클릭한다. 이 과정에서 각각의

연산자인 "="나 "?"는 콤보박스(Combo Box)형태의 인터페이스로 입력할 수 있다. 사용자가 위와 같이 입력을 하면 사용자 인터페이스 모듈에서 다음과 같은 필-인 질의가 생성된다.

판매@ape.ce.myongji.ac.kr, 고객, 이름, =, "권영일", ?, 전화번호

6.4 질의어 생성 예

사용자가 위와 같이 질의를 하면 사용자 인터페이스 모듈은 VDB 관리자에게 필-인 질의를 전달하고, VDB 관리자는 질의어 생성 모듈에게 필-인 질의를 전달하며, 질의어 생성 모듈은 전달된 정보를 SQL문의 형태로 변환한다. 이때 질의어 생성 모듈은 VDB의 "테이블 정보" 테이블에 있는 "데이터베이스 타입" 정보를 이용하여 RDB의 특징적인 기능을 이용한 질의문을 생성한다. 위의 (질의 예)에 의하여 질의어 생성 모듈에서 변환된 질의문은 다음과 같다.

```
select 전화번호 from 사원 where 이름 = "권영일";
```

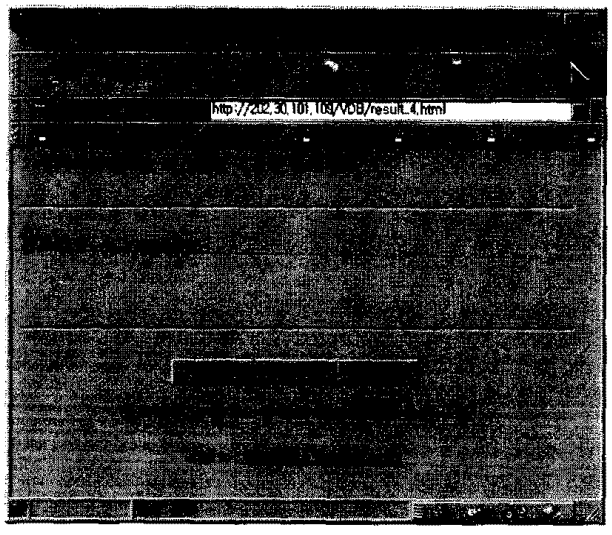
사용자가 원하는 정보를 검색하기 위한 질의문은 이것만으로 충분하나 RDB를 검색하기 위해서는 원하는 RDB를 기동(invoke)시키고 개방(open)하는 과정이 필요하다. VDB 시스템은 이러한 과정을 자동으로 처리하기 위하여 질의어 생성 모듈에서 이에 해당하는 SQL문을 생성해 준다. 또 질의어 생성 모듈은 RDB를 선택하여 연결하고 검색이 끝난 후에 RDB를 폐쇄(close)하는 과정을 위한 추가 질의문도 다음과 같이 생성한다. 이 때 "테이블 정보" 테이블에 있는 URL, 데이터베이스 이름, 데이터베이스 타입 등의 정보가 이용된다.

```
open database "판매";
select 전화번호 from 사원 where 이름 = "권영일";
close database;
```

질의어 생성 모듈은 생성된 질의문과 로그인 과정에 필요한 정보를 네트웍 모듈에 전달한다. 네트웍 모듈은 URL을 이용하여 해당하는 서버에 접속을 설정하고 RDB에 접속할 수 있는 계정으로 로그인을 한다. 네트웍 모듈은 로그인에 성공하면 질의어 생성 모듈에서 전달된 질의문을 각 DBMS에 전송하고 처리 결과를 기다린다. 로그인 과정은 각 DBMS에 따라 그 방법이 다르다. 예를 들면, 인포믹스사의 유니버셜 서버의 경우는 호스트 컴퓨터에 로그인이 되면 데이터베이스에 따로 로그인을 하지 않아도 데이터베이스를 사용할 수 있는 반면에, SQL 서버나 오라클(Oracle)의 경우는 호스트 컴퓨터에 로그인을 한후 DBMS에 다시 로그인을 해야 데이터베이스를 사용할 수 있다. 이러한 문제를 해결하기 위하여 본 논문에서는 네트웍 모듈이 서비스되는 모든 형태의 DBMS에 대한 로그인 스크립트를 만들어 사용한다. 로그인 스크립트는 네트웍 모듈이 데이터베이스 서버와 연결을 설정할 때 데이터베이스 타입 정보에 의하여 선택된다.

6.5 질의어의 처리 결과 예

질의어의 처리 결과가 생성되면 네트웍 모듈은 검색 결과를 사용자 인터페이스 모듈의 결과집합파일에 붙여(append) 쓴다. 사용자 인터페이스 모듈은 수시로 결과집합파일을 검사하여 사용자의 질의 결과가 생성되었는지 확인하고 결과가 생성되었을 경우 이를 웹을 통하여 표시하여 준다. 위의 (질의 예)에 대한 처리 결과가 웹에 표시된 화면은 (그림 6)과 같다.



(그림 6) 질의어의 처리 결과
(Fig. 6) Processing Result of Query

사용자 인터페이스 모듈은 결과집합파일을 검색하여 위에 보여준 질의 예의 결과를 찾고 그 내용을 (그림

6)에 보이는 것과 같은 HTML 파일로 만들어 화면에 표시한다. 즉 예에서 질의한 ape.ce.myongji.ac.kr에 위치한 판매 데이터베이스에 나타나는 "권영일" 이라는 고객의 전화번호는 02-3333-3333 이 된다.

7. 분석 및 결론

웹상에서 데이터베이스를 검색하는 기존의 두 가지 방법은 다음과 같은 공통적인 문제점들이 있었다. 첫째, 하나 이상의 데이터베이스를 동시에 검색할 수 없다는 점 둘째, 클라이언트 프로그램에서 미리 정의된 질의 이외의 다양한 종류의 질의를 할 수 없다는 점 셋째, 검색의 정확성이 떨어지고 불필요한 전송에 의한 네트웍 트래픽이 증가한다는 점이었다. 이러한 문제점들을 해결하기 위해서 VDB를 이용한 검색시스템을 설계하였다.

VDB 검색시스템은 위의 문제점들을 다음과 같이 해결할 수 있다.

첫째, VDB는 하나 이상의 데이터베이스 테이블 정보를 동시에 웹 화면에 표시된다. 따라서 사용자는 이 웹 화면을 보고, 하나 이상의 테이블에 각각 다른 조건으로 검색할 수 있어서 하나 이상의 데이터베이스를 동시에 검색할 수 있다.

둘째, 사용자는 웹 화면에서 필-인 방식으로 다양한 형태의 질의를 수행할 수 있으므로 직접 SQL문을 입력하는 것과 같은 효과를 얻을 수 있다. VDB의 사용자 인터페이스는 웹 화면상에서 GUI 형태로 제공되어 기존의 클라이언트 프로그램과 같이 사용이 쉬울 뿐만 아니라 융통성 있는 다양한 종류의 질의를 할 수 있다.

셋째, 기존의 방법에서는 질의어의 처리 결과가 돌아올 때까지 어떤 내용이 검색되어 돌아올지 알 수가 없으므로 불필요한 정보가 포함될 가능성이 높았다. 그러나 VDB를 사용하여 웹 화면상에 나타난 스키마(테이블) 정보를 보고 필요한 스키마에 대해서만 직접 질의를 함으로써 정확한 질의와 처리 결과에 대해 예측할 수 있다. 따라서 검색의 정확성이 향상되고 네트웍 트래픽이 감소된다.

앞으로 VDB 시스템에서 스키마 데이터의 자동 생성과 서로 다른 데이터베이스 서버에 있는 테이블간의 조인 문제를 해결한다면 유용한 데이터베이스 검색시스템이 될 것으로 기대된다.

참 고 문 헌

[1] 윤성욱, 박영배, "Web상에서 데이터베이스 검색을 위한 가상데이터베이스 생성", '97 봄 학술발표논문집(B), 한국정보과학회, 제24권 1호, pp.31-34, Apr. 1997.

[2] 이동욱, 윤성욱, 박영배, "Web상에서 가상 DB를 이용한 DB 검색 시스템", '97년 추계 학술발표논문집, 한국정보처리학회, pp.507-511, Oct. 1997.

[3] 윤성욱, "웹상에서 가상데이터베이스를 위한 사용자 인터페이스의 구현", 석사학위논문, Feb. 1998.

[4] Adam Denning, "ActiveX Controls Inside Out", Microsoft Press, 1997.

[5] M. F. Arlitt and C. L. Williamson, "Web Server Workload Characterization: The Search for Invariants", Proc. of SIGMETRICS96, ACM, Philadelphia, May, 1996.

[6] WWW3 Search Engines, http://cui.unige.ch/meta-index.html

[7] William B. Frakes and Ricardo Baeza-Yates, "Information Retrieval Data Structures and Algorithms", Prentice Hall, pp 8-12

[8] Arman Daniesh, "Teach yourself JavaScript in a week", Sams net, 1996.

[9] John Deep and Peter Holfelder, "Developing CGI Application with Perl", John Wiley & Sons, Inc., 1996.

[10] L. Latham, "Client/Server Computing: Strategic Directions, Tactical Solutions", InSide Gartner Group This Week, Vol.X, No.20, pp.1-5, Gartner Group, May 18, 1994.

[11] T. Berners-Lee, Uniform Resource Locators, Internet RFC 1738, Dec. 20, 1994 URL : ftp://ds.internic.net/rfc/rfc1738.txt

[12] K. Hughes, Entering the World-Wide Web: A Guide to Cyberspace, Honolulu Community College, Sep. 1993.

[13] T. Berners-Lee, D. Connolly, Hypertext Markup Language Specification - 2.0, Inte-

rnet RFC 1866, Nov. 1995.
[14] Pyung-chul Kim, "A Taxanomy on the Architecture of Database Gateways for Web", Internal paper of Chungnam National Univ., 1996.

이 동 욱

1971년 경북대학교 학사
1980년~1982년 ETRI SW개발 연수실장
1982년~1989년 DACOM 행정전산망 개발 본부장
1984년 정보처리 기술사
1986년 연세대학교 공학석사
1987년~1988년 하버드대학교 정보자원 정책연구소 선임연구위원 근무
1997년 명지대학교 대학원 컴퓨터공학과 박사과정 수료
1990년~현재 삼보컴퓨터, 나래시큐리티 대표이사
관심분야 : SI, MIS, 소프트웨어공학, 데이터베이스

박 영 배

1974년 동아대학교 공학사
1980년 연세대학교 공학석사
1993년 서울대학교 (컴퓨터공학) 공학박사
1974년 3월~1981년 2월 한국전력공사 (현)정보처리처과장
1990년 3월~1992년 2월 명지대학교 전자계산소 소장
1997년 2월~현재 산업대학원 원장 재직
1981년 3월~현재 명지대학교 컴퓨터공학과 교수 재직중이며, 데이터베이스, 자료구조, 화일처리 등을 강의
관심분야 : 인터넷 DB, 멀티미디어 DB, 시스템통합(SI) 등