韓國 OA學會 論文誌
第3卷 第3號, 1998. 10.

98-3-3-1-4

멀티미디어 데이터 처리를 위한 ODBC 드라이버 개발에 관한 연구

이말례* 박일록**

A Study of Development ODBC Driver for Multimedia Data Processing

Mal-Rey Lee* Il-Rok Park**

요 약

ODBC는 동적인 클라이언트/서버 환경에서 특히 효과적이다. ODBC는 멀티미디어 데이터를 처리할 수 있는 특징 때문에 멀티미디어 서버로 지원할 수 있다. 본 논문에서는 관계형 DBMS 위한 ODBC 드라이버를 개발하였다. 이 드라이버는 클라이언트 모듈과 서버 모듈로 구성되었다. 클라이언트 모듈은 SRM 이라 하고 서버모듈은 CSM 이라 하였다. 이들 두 모듈은 NSM 이라 부르는 네트워크 모듈을 통해서 연결된다. 본 논문에서 개발한 ODBC 드라이버 테스트는 기능과 정보처리 상호 운용 가능성을 테스트하였으며, 그 결과 드라이버는 클라이언트 DBMS 툴로서 성공적이었고, 멀티미디어 데이터 처리가 가능할 뿐만 아니라 DBMS 툴을 포함하는 클라이언트 응용 프로그래밍 인터페이스를 지원할 수 있는 것으로 나타났다.

Abstract

The ODBC(Open Database Connectivity) is particularly efficient in the dynamic client/server environment. Besides, the ODBC provides the functions that can handle multimedia data. Due to this feature, we are able to use the DBMS that supports the ODBC as a multimedia server.

In this thesis, we describe the development of the ODBC driver for Relational DBMS. The Relational DBMS ODBC Driver consists of the client module and the server module. The client module is called the SRM(Server Request Module) and the server module is CSM(Client Service Module). These two modules are connected through the network module called the NSM(Network Service Module).

We have conducted both the functional and the interoperability test of our ODBC Driver. It turned out that the ODBC driver operated with these client DBMS tools successfully. In all, due to our development of the Relational DBMS ODBC Driver, DBMS is now capable of processing multimedia data and supporting the client applications including the DBMS tools.

* 조선공업대학 교양 전산학 전임강사
* 조선공업대학 전자계산소
논문접수:98.8.28. 심사완료:98.10.30.

Ⅰ. 서론

데이터베이스 시스템을 데이터 서버로 활용하기 위해서는 응용 프로그램에게 데이터 서버에 접근할 수 있도록 하는 인터페이스가 제공되어야 한다. 즉, 이러한 응용 프로그래밍 인터페이스(API)를 통해서 응용 프로그램이 작성되며 따라서 데이터베이스 시스템을 데이터 서버로 사용할 수 있게 된다.

이러한 응용 프로그래밍 인터페이스로 그 동안 가장 널리 사용되어온 것은 Embedded SQL(ESQL)이다. ESQL로 작성된 응용 프로그래밍은 프로그램 내 SQL 명령을 target DBMS 환경에 의존적이며, 따라서 다른 데이터 서버를 target으로 데이터 처리를 할 수 없다. 이로 인해 ESQL은 현재 일반적인 컴퓨팅 환경인 클라이언트/서버 구조와 같은 동적인 환경에는 적합하지 못하다. 이러한 한계를 극복하기 위해 등장한 데이터베이스 API가 SQL Call Level Interface(CLI)이다. CLI는 응용 프로그램이 여러 target DBMS들과 실행 모듈 수준의 호환성(binary compatibility)을 갖도록 하여 DBMS 서버와 클라이언트 응용 프로그램 및 툴들 간의 연동(interoperability)을 효율적으로 제공한다[1].

SQL CLI에는 X/Open 과 SQL Access Group에서 발표한 표준 사양[2], ISO/ANSI에서 발표한 표준 사양, Microsoft사가 발표한 ODBC는 X/Open 및 ISO의 표준 사양을 확장한 것으로 멀티미디어 데이터 처리를 지원하며 현재 업계 표준 PC 클라이언트 API이다.

ODBC는 응용 프로그램에 인터페이스 함수의 형태로 제시되는데 이 인터페이스 함수는 모두 55개로서 각 함수들은 ODBC 사양에서 분류한 방식에 따라 나뉘게 된다. 이 분류 기준을 ODBC API Conformance level 이라 하는데 이에 따라 각 인터페이스 함수들은 Core Level, Level 1, Level 2로 나뉘게 된다. 이러한 ODBC 인터페이스 함수로 작성된 응용프로그램은 target DBMS에 독립적인 특성을 가진다. 즉, ODBC 인터페이스로 작성된 응용 프로그램의 실행 모듈은 ODBC를 지원하는 각 DBMS 에 접근하여 원시코드의 수정 혹은 재 컴파일 없이 이를 데이터 서버로 활용할 수 있게 된다.

여기서 각 DBMS 가 ODBC를 지원한다는 것은 ODBC 인터페이스로 작성된 응용 프로그램이 실행 시에 호출하는 ODBC 인터페이스 함수에 대해서 각 DBMS 의 SQL 처리기의 기능을 호출할 수 있는 라이브러리를 각 DBMS 가 갖추고 있어야 함을 의미한다. 이러한 라이브러리를 각 DBMS 의 ODBC 드라이버라 칭하며 이 ODBC 드라이버는 응용 프로그램의 실행 시에 동적으로 링크, 적재되는 동적 링킹 라이브러리로 제공된다. 이러한 ODBC 드라이버를 갖추고 있는 다양한 DBMS 에 대해서 ODBC 인터페이스로 작성된 응용 프로그램은 ODBC 사양에 따라 각 DBMS에 독립적으로 작성될 수 있으며 실행에 있어서도 응용 프로그램의 실행 모듈 그 자체로 여러 DBMS 와 호환성을 지니며 실행될 수 있다.

이러한 ODBC의 장점 이외에도 ODBC가 각광받는 것은 많은 클라이언트 응용 소프트웨어들이 ODBC를 지원하는 현재의 추세에 기인한다. 즉, 현재 많은 상용 클라이언트 DBMS 툴들이 ODBC를 지원, 툴의 사용에 있어서 ODBC 인터페이스를 통하여 데이터베이스 서버에 접근할 수 있는 기능을 제공하고 있으며 대표적인 DBMS 벤더들은 모두 자사의 DBMS 엔진을 위한 ODBC 드라이버를 제공하고 있다[3].

본 논문에서는 클라이언트/서버 DBMS 인 관계형 DBMS를 위한 ODBC 드라이버의 개발해 대하여 기술한다. 관계형 DBMS 는 LOB(Large OBject) 데이터형을 제공하는 멀티미디어 데이터 서버로서[5], 본 논문에서 기술하는 ODBC 드라이버의 개발을 통하여 망 환경에서 PC 클라이언트들을 지원할 수 있게 되었다. 따라서 본 논문의 연구는 다음과 같은 순으로 수행하였다.

- 첫째, 클라이언트/서버 DBMS 환경에서 ODBC 인터페이스를 지원하는 ODBC 드라이버를 설계 및 구현하였다. 구현한 ODBC 드라이버는 관계형 DBMS 에 대한 ODBC 드라이버를 클라이언트 PC의 MS Windows 95 환경에서 구현하였다.

둘째, ODBC를 지원하는 상용화된 많은 DBMS 툴들에서 구현한 ODBC 드라이버를 호출하여 데이터베이

스 서버에 접근하는 연동 검사를 통하여 구현한 ODBC 드라이버가 정상적으로 ODBC 인터페이스를 지원하는지 여부를 검사하였다.

II. ODBC의 개념과 구성

ODBC는 마이크로 소프트사가 발표한 데이터베이스 API이다. ODBC의 구성은 〈그림 2.1〉에서 제시하는바와 같이 응용 프로그램, ODBC 드라이버 관리자, ODBC 드라이버, 데이터 소스로 이루어져 있다.

〈그림 2.1〉에서 응용 프로그램은 ODBC 인터페이스 함수의 호출 형태로 작성되고 실행시 이 호출을 ODBC 드라이버 관리자로 보낸다. ODBC 드라이버 관리자는 마이크로 소프트사에서 제공하는 부분으로서 각 ODBC 드라이버를 적재시키는 주된 역할을 수행하며 동적 링킹 라이브러리로 제공된다. 또한 ODBC 드라이버 관리자에 의해 적재되며 ODBC 인터페이스가 실제 코드로 구현된 부분인 ODBC 드라이버는 각 데이터 소스별로 제공되며, ODBC 구성상 가장 하부에 위치한 데이터 소스는 각 DBMS 와 기타 제반 사항을 통합한 하나의 인스턴스의 개념으로 각 데이터베이스를 대표한다.

ODBC 구성 요소 중 본 논문에서의 개발 사항은 ODBC 드라이버 부분이다. 즉, ODBC 드라이버를 구현하여 구현한 드라이버가 〈그림 2.1〉 상의 여러 ODBC 드라이버와 마찬가지로 구성 요소로 연결될 수 있도록 하는 것이 본 논문의 연구 내용이다.

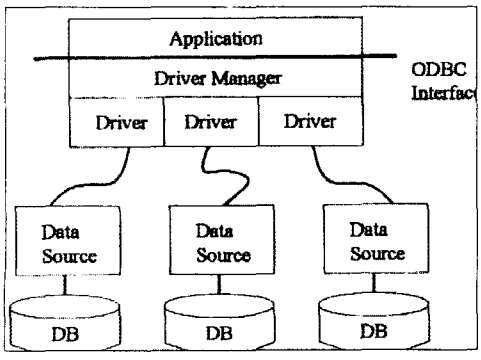

그림 2.1 ODBC의 구성

III. ODBC 드라이버의 개발

본 논문의 연구에서는 ODBC 드라이버 구현을 관계형 DBMS 에 적용하여 구현하였다.

3.1 구 조

ODBC의 구성을 나타내는 〈그림 2.1〉에서 ODBC 드라이버 부분을 구현하기 위해서는 이 ODBC 드라이버 부분이 제공하는 기능을 모듈로 나누어 구현하는 것이 효율적이다. 이것은 클라이언트/서버 DBMS 환경에서 ODBC 드라이버가 지원해야 하는 기능을 클라이언트 측과 서버 측으로 나누어 제공하기 위해 더욱 필요하다.

따라서 본 논문에서는 관계형 DBMS ODBC 드라이버의 구조를 〈그림 3.1〉과 같이 설계하였다.

관계형 DBMS ODBC 드라이버는 크게 SRM (Server Request Module), CSM(Client Service Module) 그리고 이들 두 모듈을 망을 통해 연결하는 NSM(Network Service Module)로 구성된다. SRM은 클라이언트측 모듈로서 응용 프로그램(AP)이 호출한 ODBC 인터페이스 함수의 데이터 서비스 요청을 관계형 DBMS가 인식할 수 있는 형태로 변환하여 이를 서버측에 요청하고 그 결과를 AP에 반환하는 역할을 수행한다. CSM은 서버측 모듈로서 SRM의 요청을 관계형 데이터베이스에 접근하여 처리하는 모듈이다. NSM은 관계형 DBMS ODBC 드라이버의 구현 환경이 PC 환경의 클라이언트와 UNIX 환경의 DBMS 서버로 구성된 구조이므로 이들을 망으로 연결하기 위해 클라이언트 및 서버측에 각각 존재하며 SRM 및 CSM 의 라이브러리로 구성된다. 그리고 클라이언트 연결 데몬이란 데몬 프로세스를 구동하여 SRM과 CSM 간의 연결을 관리하도록 하였다.

AP의 관계형 DBMS 접속 과정은 AP의 실행이 시작되어 관계형 DBMS 서버와 연결을 시도하면 AP는 SRM과 동적으로 링크되고 클라이언트 연결 데몬에 의해 서버측의 CSM과 연결된다. 이 연결은 망을 통한

것으로서 socket이 사용된다. 그후 CSM은 관계형 DBMS 엔진의 프로세스 구조에 따라서 데몬 프로세스인 연결 데몬에 의해 관계형 DBMS 엔진 프로세스와 메시지 큐를 통해 연결된다.

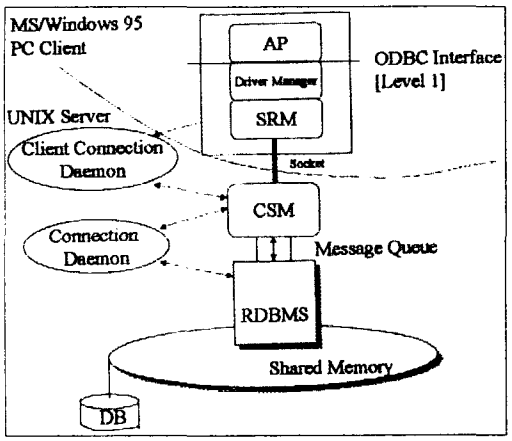

그림 3.1. OBDC 드라이버의 구현 구조

3.2 구 현

관계형 DBMS ODBC 드라이버의 구현 내용은 클라언트측의 SRM과 서버측의 CSM, 그리고 NSM의 구현을 기술한다.

3.2.1 SRM

관계형 DBMS ODBC 드라이버의 구현을 위한 클라이언트 측 모듈인 SRM(Server Request Module)의 구현을 위해 먼저 구현에 필요한 자료 구조를 클래스 구조로 정의하였다. 구성한 클래스는 각각의 멤버 함수를 통해 서로 연결되며 특정한 클래스로부터 상속받아 새로운 클래스를 구성하기도 한다. 이와 같은 클래스간의 연결과 생성을 통해서 ODBC 드라이버를 구성하는 인터페이스 함수들을 구현하였다.

SRM을 구현하기 위해 클래스를 구성한 후 구성한 클래스를 사용하여 API Conformance Level 의 Core, Level1에 해당하는 총 38개의 ODBC 인터페이스 함수를 구현하였다. 각 ODBC 인터페이스 함수의 구현은 각 인터페이스 함수가 코드 형태의 유사성을 가지고 있다. 이것은 각 ODBC 인터페이스 함수가 수행하는 기능은 다르지만 각 ODBC 인터페이스 함수는 호출되었을 때 일반적으로 제공해야 하는 기능을 공통적으로 가지고 있으므로 대략적인 함수 내부의 구현 형태는 비슷하다는 것이다. 〈그림 3.2〉는 이러한 ODBC 인터페이스 함수의 일반적인 구현 Pseudo Code를 나타낸다.

```
extern "C" RERCODE SQL_API_export
  SQL* (.....)
{
각 핸들에 대한 정당성 검사
함수 호출로 인한 state transition 검사
ODBC_CSM 함수 호출
ODBC_CSM 으로 부터 반환 받은 값을 검사
  하여 에러 판단 리턴 값 반환
}

(SQL* 는 임의에 ODBC 인터페이스 함수
```

그림 3.2. ODBC 인터페이스 함수의 일반적인 구현

관계형 DBMS ODBC 드라이버의 클라이언트 측 구현 모듈인 SRM은 MS Window 95 환경에서 Visual C++ 5.0로 구현하였으며 동적 링킹 라이브러리 형태로 작성하였다.

3.2.2 CSM(Client Service Module)

클라이언트 측 모듈인 SRM으로부터 호출되며 데이터 소스로부터 서비스를 제공받아 다시 SRM으로 반환하는 역할을 수행하는 CSM은 UNIX 환경에서 관계형 DBMS의 ESQL/C로 구현하였다. 즉, SRM에 제공하는 CSM 인터페이스를 정의하고 그 인터페이스에 해당하는 함수를 ESQL/C 프로그래밍으로 구현한 것이다. CSM 인터페이스는 24개를 정의하였으며 24개의 CSM 인터페이스 함수는 CSM 인터페이스 함수가 제공하는 기능에 따라 각각 SRM 내의 ODBC 인터페이스 함수 내부에서 호출된다. 즉, 클라이언트 ODBC 인터페이스 함수에서 CSM 인터페이스 함수를 호출하게 된다. 이후 CSM 인터페이스 함수는 내부에 구성된 ESQL/C 구문으로 데이터 소스에 접근하게 되고 다시 데이터 소스로부터 제공받은 서비스의 결과를 NSM을 통하여 클라이언트 측의 ODBC 인터페이스 함수로 반환하게 된다.

3.2.3 NSM

SRM 과 CSM의 통신을 위한 NSM(Network Service Module)은 SRM 과 CSM에 각각 라이브러리의 형태로 제공된다. NSM의 개발 환경은 SRM 에서는 Window 95에서 제공되는 Winsock을 사용하고 CSM에서는 Berkeley socket을 사용하여 socket 프로그래밍으로 구현하였다.

본 논문에서 사용하는 socket 함수들은 데이터를 전송할 때 문자형 데이터의 전송만을 허용하므로 SRM과 CSM간에 전송되는 모든 데이터는 문자형으로 반환되어 버퍼에 저장되어 전송되는 방법을 사용하였다. 전송을 위한 버퍼의 내부에는 각 필드를 나누어 SRM 과 CSM에서 각 필드의 의미를 공유하여 전송시 각 필드를 해석하여 전송되는 데이터를 얻어내게 된다.

3.3 기능 테스트

관계형 DBMS ODBC 드라이버에 대한 기능 테스트는 검사하고자 하는 항목별 응용 프로그램으로 구성된 테스트 수트를 작성하여 수행하였다. 즉, 이 방법은 구현한 ODBC 드라이버를 총괄적으로 테스트하기 위한 방법으로서 테스트를 요하는 각 항목을 작성하여 이를 각각 응용 프로그래밍으로 작성하고 이를 간략히 수트화하여 이 수트를 일괄적으로 실행함으로써 구현한 ODBC 드라이버의 기능을 테스트하는 것이다.

작성한 테스트 수트는 정상 작동을 검사하는 항목 뿐 아니라 에러 상황의 처리를 검사하는 항목도 포함하고 있으며 정형 데이터의 처리 및 멀티미디어 데이터의 처리를 검사하는 내용으로 구성되어 있다. 테스트 내용은 다음과 같다.

3.3.1 정형 데이터를 처리하는 수트 테스트

정형 데이터를 처리하는 수트는 검사하고자 하는 항목을 함수 모듈로 각각 작성하여 통합한 하나의 응용 프로그램으로 작성되어 있다. 이 응용 프로그램 내의 테스트 항목의 실행은 main()에서 각 테스트 항목에 해당하는 함수를 각각 호출하면서 그 결과를 결과 파일에 작성한다. 정형 데이터를 처리하는 수트의 수행 결과를 확인하는 방법은 결과 파일을 확인하여 각 테스트 항목별로 실행 결과가 pass 인지를 확인하여 모든 테스트 항목의 결과가 pass 인지를 확인하였다.

정형 데이터를 처리하는 수트는 모두 19개의 항목으로 구성되어 있는데 각 항목은 정상 기능을 검사하는 항목과 함께 에러가 발생하는 상황을 유발시켜서 정확한 에러 처리 여부를 분석하는 항목으로 구성되어 있다. 〈그림 3.3〉는 정형 데이터를 처리하는 수트의 항목과 결과가 모두 pass 하였음을 나타낸다.

번호	테스트 항목	테스트 함수	결과
1	핸들할당, 정상DB 연결, 연결 중지, 핸들해제	Alloc*(), Connect(), Disconnect(), Free*()	pass
1_1	SQLDriverConnect()를 통한 DB 연결	DriverConnect()	pass
1_err1	연결전 SQLAllocSmt()호출	AllocSmt()	pass
1_err2	연결전 SQLDisconnect()호출	Disconnect()	pass
2	테이블 생성	Prepare(), Execute()	pass
2_err1	테이블 생성시 이미 존재하는 테이블 이름 사용	Execute()	pass
2_err2	잘못된 입력 인자의 사용	Prepare()	pass
3	정형 데이터 입력	BindParameter(), Execute()	pass
3_1	SQLPutData()를 이용한 정형 데이터 입력	BindParameter(), ParamData(), PutData(), Execute()	pass
3_err1	입력갯수와 테이블 필드수가 맞지 않음	Execute()	pass
3_err2	바인딩된 매개변수의 수와 SQL구문의 매개변수 수가 다름	Execute()	pass
3_err3	SQLBindParameter()에 잘못된 입력 인자의 사용	BindParameter()	pass
4	SQLFetch(), SQLNumResultCols()를 이용한 데이터 검색	BindCol(), NumResultCols(), Fetch(), DescribeCol()	pass
4_1	비 문자형 데이터 검색	ExecDirect(), GetData(), ColAttributes()	pass
4_2	문자형 데이터 검색	ExecDirect(), GetData()	pass
4_err1	커서가 오픈된 상태에서 Prepare()	Prepare()	pass
4_err2	SQLBindCol() 호출시 잘못된 컬럼번호	BindCol()	pass
5_err1	다른테이블 실행시 add절에 명시된 필드가 이미 존재	Execute()	pass
5_err2	create index 실행시 기본 테이블이 존재하지 않음	Execute()	pass

그림 3.3. 정형 데이터를 처리하는 수트의 항목과 결과

3.3.2 멀티미디어 데이터를 처리하는 수트 테스트

멀티미디어 데이터를 처리하는 수트는 ODBC 인터페이스 함수 중 멀티미디어 데이터의 처리를 지원하는 함수를 검사하기 위한 수트로 정형 데이터를 처리하는 수트와 마찬가지로 멀티미디어 데이터를 처리하는 함수에 대해 검사하고자 하는 항목을 먼저 작성하여 각각 함수로 구성하고 이를 하나의 응용 프로그램으로 작성하였다.

테스트 수트가 실행되는 순서는 응용 프로그램 내 main()에서 각 테스트 항목에 대응하는 함수를 호출하는 순서를 따른다. 테스트 결과를 확인하는 방법 역시 응용 프로그램에서 생성하는 결과 파일을 통해 각 테스트 항목이 모두 pass로 출력되었는지를 검사한다.

멀티미디어 데이터를 처리하는 수트의 실행 결과는 정형 데이터를 처리하는 수트의 결과를 판단하는 기준인 ODBC 인터페이스 함수의 반환 값만으로 판단하기에는 불충분하다. 따라서 테스트 항목이 수행되기 전 각 항목마다 예상 결과 파일을 준비해 두고 각 테스트 항목이 수행된 후 생성된 파일과 테스트 한 후 생성된 파일의 비교는 응용 프로그램 내부에 두 파일을 비교하는 함수를 따로 작성하여 이 함수를 호출함으로서 비교하였다. 멀티미디어 데이터를 처리하는 수트는 모두 5개의 항목으로 구성되어 있다. 각 테스트 항목과 결과 〈그림 3.4〉와 같다.

번호	테스트 항목	테스트 함수	결과
1	멀티미디어 데이터 필드를 가지고 있는 테이블 생성	Prepare(), Execute()	pass
2	PutData(), ParamData()를 이용한 멀티미디어 데이터 입력	PutData(), ParamData(), BindParameter()	pass
3	GetData()를 이용한 멀티미디어 데이터 검색	GetData()	pass
4	테이블내 튜플 삭제	ExecDirect(), RowCount()	pass
5	테이블 삭제	Execute(), Transact()	pass

그림 3.4. 멀티미디어 데이터를 처리하는 수트의 항목과 결과

Ⅳ. 연동을 이용한 응용

본 논문에서는 구현한 ODBC 드라이버 연동 검사를 보다 완전히 하고 연동 가능의 성과를 제시하기 위하여 클라이언트 응용 개발 툴 중 하나인 Powerbuilder 5.0에서 클라이언트 응용 프로그램을 개발해 보았다. 따라서 본 장에서는 먼저 개발한 클라이언트 응용 프로그램을 소개하고 클라이언트 응용 프로그램을 구성하며 실행하는 순으로 설명한다.

4.1 클라이언트 응용 프로그램

Powerbuilder 5.0을 통해 개발한 클라이언트 응용 프로그램은 도서 명세를 관리하는 것으로서 도서의 입력, 검색, 삭제, 갱신 등의 일반적인 데이터에 대한 연산을 모두 제공한다.

본 연구에서 개발한 클라이언트 응용 프로그램의 실행은 Powerbuilder 5.0 내 실행 메뉴를 선택하여 수행된다. 클라이언트 응용 프로그램이 실행되면 먼저 DBMS 데이터 소스에 연결된다. 그리고 클라이언트 응용 프로그램의 메인이 화면에 나타난다. 실행하는 응용 프로그램이 지원하는 여러 기능은 메인 윈도우에 있는 5개의 버튼인 입력, 조회, 삭제, 저장, 종료가 각각 담당한다. 즉, 입력 버튼을 통해서 도서관리 테이블 내에 튜플을 입력할 수 있으며, 조회 버튼을 통해서 현재 도서관리 테이블 내에 있는 튜플을 선택하여 검색할 수 있다. 또한 삭제 버튼을 통해서는 현재 실행 윈도우에 제시되고 있는 튜플을 삭제할 수 있으며, 저장 버튼을 통해서는 현재 실행 윈도우에 제시되고 있는 튜플에 대해 변경하고자 하는 컬럼을 선택하여 사용자가 입력하는 값으로 변경하게 된다. 마지막으로 종료 버튼을 통해서는 연결되어 있던 데이터베이스와 연결을 해제하고 응용 프로그램을 종료하게 된다. 따라서 본 논문의 기술에서는 이러한 실행에 대한 설명과 실행 항목의 순서대로 제시한다.

4.1.1 입력의 실행

메인 화면의 입력 버튼을 선택하면 화면에 튜플의 입력을 위한 윈도우가 생성된다. 이 윈도우의 해당 칸을 작성하여 삽입 버튼을 선택하면 도서관리 테이블에 입력이 실행된다. 〈그림 4.1〉 이러한 입력을 위해 생성되는 윈도우를 나타낸다.

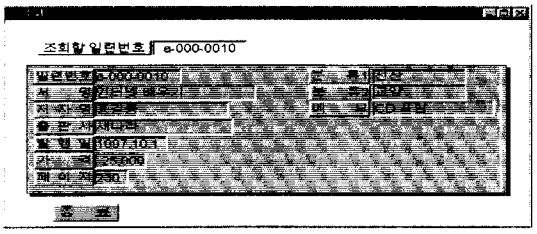

그림 4.1. 데이터 입력 윈도우

4.1.2 검색의 실행

〈그림 4.2〉는 실행 윈도우의 검색 버튼을 선택하였을 때 선택한 튜플을 검색한 상황을 나타낸다. 즉, 검색 버튼을 선택하여 검색하고자 하는 튜플을 선택하고 이를 검색한 결과를 화면의 해당 컬럼을 채워 나타내게 된다. 즉 〈그림 4.2〉는 앞서 입력한 튜플을 검색한 결과이다.

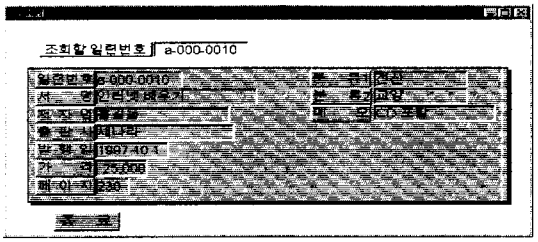

그림 4.2. 데이터 검색

4.1.3 삭제의 실행

메인 윈도우의 삭제 버튼을 선택하여 현재 화면에 제시된 튜플을 삭제할 수 있으며 삭제가 실행되면 실행의 성공을 나타내는 윈도우가 나타난다.

4.1.4 갱신의 실행

갱신의 대상은 현재의 메인 윈도우상에 제시된 튜플이며 갱신하고자 하는 컬럼의 선택을 위해서 윈도우가 제시되며 이 윈도우에서 선택한 컬럼에 대해 갱신하고자 하는 값을 입력 받아 실제 튜플에 이를 반영한다. 〈그림 4.3〉은 갱신하고자 하는 대상 컬럼을 선택하는 윈도우를 나타내며 〈그림 4.4〉는 〈그림 4.3〉에서 name 컬럼에 해당하는 서명 항목을 선택하여 갱신하고자 하는 값을 입력하는 윈도우이다.

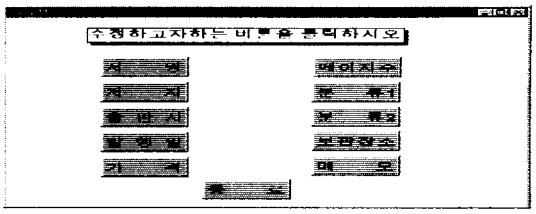

그림 4.3. 갱신을 위한 컬럼 선택 윈도우

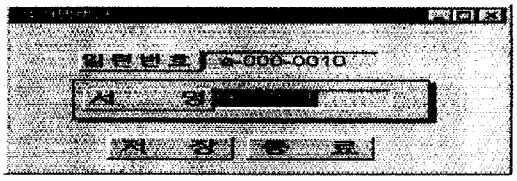

그림 4.4. 갱신 데이터 입력 윈도우

또한 〈그림 4.5〉는 갱신한 후 다시 조회를 한 상황을 나타낸다. 〈그림 4.5〉의 상황은 〈그림 4.2〉와 비교할 때 name 컬럼인 서명 항목의 값이 갱신되었음을 알 수 있다.

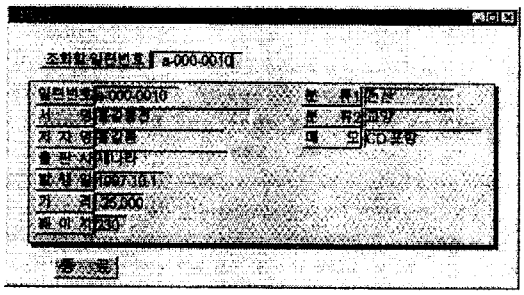

그림 4.5. 갱신 후 데이터 검색

4.1.5 평가

Powerbuilder 5.0에서 개선된 ODBC 드라이버와 연동을 이용하여 작성한 응용 프로그램은 그 구동에 있어서 정상적인 실행을 나타냄으로써 관계형 DBMS ODBC 드라이버의 연동 가능을 확인하였다.

V. 결 론

본 논문은 관계형 DBMS를 위한 ODBC 드라이버의 개발 결과를 기술하였다. ODBC 인터페이스는 클라이언트 응용 프로그래밍 인터페이스로서 동적인 클라이언트/서버 환경에 적합하다. 또한 ODBC는 멀티미디어 데이터 처리를 지원하는 클라이언트 응용 프로그래밍 인터페이스로서 상용화된 대표적인 DBMS 들이 모두 이를 지원하고 있다.

본 논문에서는 관계형 DBMS ODBC 드라이버를 MS/ODBC 사양에서 제시하는 인터페이스 Conformance 수준을 인터페이스 Conformance 수준1까지 지원하도록 구현하였다. 그리고 구현한 관계형 DBMS ODBC 드라이버의 기능을 검사하기 위하여 테스트 수트를 이용한 기능 테스트를 수행하였으며 다양한 DBMS 툴과 연동 테스트를 수행하였다. 기능 테스트와 연동 테스트 결과 구현한 관계형 DBMS ODBC 드라이버는 두 테스트를 모두 통과하였다. 이러한 드라이버 개발은 멀티미디어 데이터 처리를 지원하는 응용 프로그래밍 인터페이스를 지원하므로 클라이언트/서버 환경에서 멀티미디어 서버로 기능을 담당할 수 있게 되었다. 또한 ODBC를 지원하는 다양한 DBMS 툴과 연동이 가능하므로 툴의 사용에 있어서 ODBC 인터페이스를 통하여 데이터베이스 서버에 접근할 수 있는 기능을 제공하게 되었다.

참고문헌

[1]. Dunja Mladenic , "ISO-ANSI(Working Draft) SQL Call-Level-Interface(CLI)," ISO DBL MUN-005/ANSI X3H2-93-360, 1993.

[2]. David Ad and Oren Bar-Ner, "Microsoft ODBC 2.0 Programmer's Reference and SDK Guid," Microsoft Corp, 1994.

[3]. K. North, "Understanding Multidatabase APIs and ODBC," DBMS Vol.7, No. 3, pp. 44-55, 1994.

[4]. Chung T.Kwok, "Getting Started Building Applications Quick Reference," Powersoft Corp, 1994.

[5]. Robert C. Holte, "Data Management: SQL Call Level Interface(CLI)," X/Open Company Ltd Snapshot. 1995.

저 자 소 개

이말례

1997년 중앙대학교 컴퓨터공학과 박사졸업

1998.3-1998년 10월 현재 조선공업대학 교양 전산학 전임강사

관심분야: 유전자 알고리즘, 전문가 시스템, 학습, 신경망, CAI, 멀티미디어 인공생명 등

박일록

1997년 조선대학교 산업대학원 전자계산학과 석사졸업

1994. 1- 1998. 10월 현재 조선이공대학 전자계산소 근무

관심분야: 인공지능, 멀티미디어, CAI 등