DataMiner로 데이터베이스연결하기

데이터베이스를 사용하는 이유

데이터베이스는 우리가 저장/조회/업데이트하고자 하는 모든 데이터들을 논리적/효율적으로 저장하는 저장소입니다. 이러한 데이터베이스를 개발하기 위해서 데이터베이스 관리 시스템을 사용하는데, 최소한 이름은 들어 보았을 Oracle, MySQL, 등이 바로 데이터베이스 관리 시스템의 하나입니다.

데이터베이스는 요즘의 어느 정도 규모가 있는 개발에서 필수적인 요소로 사용되고 있기 때문에, 자세한 내용은 시중의 다른 서적을 공부해 보는 것이 좋을 것입니다. 데이터베이스와 관련된 책은 평생 다 읽을 수도 없을 만큼 많이 나와 있습니다.

IDL 프로그래머로서 데이터베이스를 사용할 때 느끼는 장점은 다음과 같은 것입니다.

1. 다수의 사용자들이 동시에 사용하는 중앙 데이터 저장소로서의 활용성
2. 어떤 프로그래밍 언어든지 데이터베이스와 연동하는 방법이 존재함. 그러므로, 중앙에 데이터베이스가 존재하고 IDL, C++, JAVA, Visual Basic등 다른 모든 언어가 데이터베이스를 통해 연결됨.
3. 대용량 데이터에 대한 입력, 조회, 삭제, 변경 등의 표준적 방법을 제시함으로써, 프로그램 개발의 기간을 단축하고, 개발된 프로그램의 유지 보수가 용이해 짐.

IDL의 DataMiner

IDL을 데이터베이스와 연동하기 위해서 DataMiner라는 모듈을 사용하게 됩니다. 이는 IDL 설치 단계에서 IDL Dataminer(ODBC Drivers)를 선택하게 되면 함께 설치가 됩니다.

DataMiner는 ODBC(Open DataBase Connectivity)라는 데이터베이스 접근 방법을 사용하며 Data Direct사의 ODBC 드라이버를 사용합니다. 데이터베이스 관리 시스템 제조사가 제공하는 클라이언트 소프트웨어에 의존하는 일반적인 ODBC 인터페이스도 사용가능하지만, Wire Protocol(WP라는 말이 등장하면 모두 Wire Protocol을 의미합니다) 인터페이스를 선택할 수 있으며, 별도의 클라이언트 소프트웨어 없이 동작합니다.

현재 여러분의 IDL에서 DataMiner를 사용할 수 있는 상태인지 확인해 보려면 DB_EXISTS() 함수를 사용합니다.

```
IDL> print, db_exists()
% Loaded DLM: DATAMINER
      1
```

SQL 문법

데이터베이스 표준 언어를 SQL(Structured Query Language)이라고 합니다. 모든 데이터베이스가 거의 똑같은 문법을 사용하므로 사용자는 MySQL을 사용하든 Oracle을 사용하든 개념만 같다면 같은 코딩을 하게 됩니다.

이 문서는 사용자가 데이터베이스와 SQL 문법을 어느 정도 아는 선에서 IDL과 DB를 연동하기 위한 방법을 소개하고 있습니다. 데이터베이스와 SQL에 관심이 있다면, 'Head First SQL'과 같은 개념서를 한번 훑어 보세요

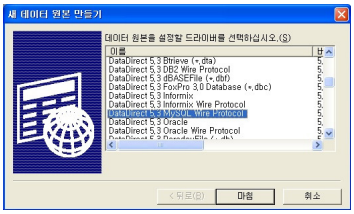
데이터베이스 연결 설정 (MS-Windows)

Windows 운영체제에서 ODBC 드라이버를 관리하는 곳은 "시작→제어판"에서 "관리도구" 폴더의 "데이터원본(ODBC)" 아이콘으로 접근합니다. 이곳은 현재 운영체제가 기억하고 있는 데이터베이스들의 목록입니다. 이미 등록된 데이터베이스가 있을 수도 있고, 처음 연결이라면 연결할 데이터베이스를 추가하면 됩니다.

1) ODBC 데이터 원본 관리자에서 추가 버튼을 클릭하세요.

2) 새 데이터 원본 만들기에서 연결하고자 하는 데이터베이스의 제조사와 관련된 드라이버를 선택합니다.

- DataDirect 로 시작하는 드라이버는 IDL의 DataMiner가 제공하는 표준 드라이버입니다만, ODBC가 원래 그렇듯, 데이터베이스 제조사가 제공하는 ODBC 드라이버로도 잘 작동합니다. MS-Access와 같은 데이터베이스는 공식적으로 IDL DataMiner가 지원하지 않지만, Office를 설치할 때 함께 설치되는 ODBC 드라이버를 이용하여 IDL과 연동이 가능합니다.
- 동일한 데이터베이스 시스템에 Wire Protocol이 붙은 드라이버가 함께 제공되는 것이 있습니다. 데이터베이스 제조사의 클라이언트 소프트웨어 없이 직접 연결하고자 하는 경우에는 Wire Protocol 쪽을 선택합니다(속도도 WP가 더 빠릅니다).



MySQL Wire Protocol 선택 예

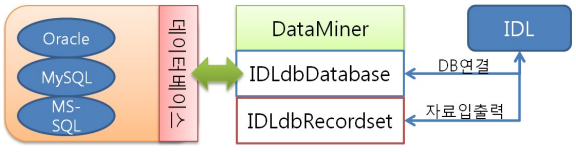
3) 선택된 제조사별 드라이버 설정 창이 열립니다. 보통의 경우 General 탭만 설정해 주면 충분합니다. 제조사별로 조금씩 다를 수 있지만 다음과 같은 내용입니다.

- Data Source Name : 운영체제에서 인식할 데이터베이스 이름입니다. 데이터베이스가 어떤 것이건 간에 설정 이후에는 이 이름만으로 데이터베이스를 찾아갑니다.
- Description : 추후 기억을 돕기 위한 설명을 써 놓으면 됩니다. 실제 프로그램의 작동과는 무관합니다.
- Host Name : 데이터베이스를 운영하는 서버쪽의 IP 또는 호스트 이름을 써 주면 됩니다.
- Port Number : 데이터베이스 원격 접속을 위해 데이터베이스 운영자가 열어놓은 포트를 의미합니다(예; 3306).
- Database Name : 하나의 데이터베이스 서버가 동시에 여러 종류의 데이터베이스를 서비스할 수 있습니다. 연결 대상 데이터베이스를 써 줍니다.

데이터베이스 연결 설정 (Linux/Unix)

리눅스와 유닉스 계열에서 DataMiner 연결은 .odbc.ini 파일을 이용합니다. IDL 설치디렉토리/resource/dm/운영체제종류/odbc.ini 파일이 있을 것입니다. 이는 기본 파일이며 이를 사용자별 계정의 홈 디렉토리/.odbc.ini 파일로 복사하고 이를 각자 수정하여 사용합니다(.odbc.ini로 맨 앞에 점(.)이 있는 것을 주의하세요). 수정할 내용은 MS-Windows용 접속과 유사합니다. Data source name이 맨 윗줄에 [ ] 안에 입력된다는 점만 유의하면 될 것 같습니다.

```
$ cp /usr/local/itt/idl71/resource/dm/linux.x86/odbc.ini ~/.odbc.ini
```



데이터 마이너의 두 클래스

IDLdbDatabase

데이터베이스와의 연결과 테이블 생성, 삭제 등의 관리를 담당합니다.

IDLdbRecordset

테이블 또는 테이블 안의 선택 레코드에 접근하는 객체를 생성합니다.

데이터 베이스 연결

```
IDL> oDB=obj_new('IDLdbDatabase')
IDL> oDB-> Connect, datasource='firstdm', $
       user_id='dmtester', password='abcdefg'
IDL> print, obj_valid(oDB)
  1
```

데이터베이스는 처음 연결에 많은 시간이 소요됩니다. 그러므로 데이터베이스를 사용할 때마다 연결을 다시하는 것은 아주 나쁜 방법이며, 위 예에서 oDB 변수(데이터베이스를 가리키는 레퍼런스)를 계속 사용하는 것이 효율적입니다.

테이블 생성과 제거

다음과 같은 테이블을 만들어 볼 예정입니다.

name	dist	period	mass	nsat
Mercury	0.3871	0.24085	0.0553	0
Venus	0.7233	0.61521	0.8149	0
Earth	1.0000	1.00000	1.0000	1
Mars	1.5237	1.88089	0.1074	2
Jupiter	5.2028	11.8623	317.938	16
Saturn	9.5388	29.458	95.181	18
Uranus	19.1914	84.0139	14.531	21
Neptune	30.0611	164.793	17.135	8
Pluto	39.5294	248.54	0.0022	1

planet 테이블의 구조

```
IDL> oDB->ExecuteSQL, "create table planets "+$
       "(name char(8) not null primary key, "+$
       "dist float, period float, mass float, nsat integer)"
```

위 문장은 IDLdbDatabase의 메쏘드인 ExecuteSQL 프로시저를 이용하여 데이터베이스 테이블을 생성하는 SQL문을 실행시킨 것이며, 'Create table planets (name char(8) not null primary key, dist float, period float, mass float, nsat integer)"는 데이터베이스 시스템에서 실행되는 SQL문입니다.

테이블을 제거하는 SQL문은 'drop table'입니다.

```
IDL> oDB->ExecuteSQL, "drop table planets"
```

이후 계속 planets 테이블을 이용할 것이므로 다시 create table 문을 실행해 두십시오. 데이터베이스 안에 테이블은 필요한 만큼 얼마든지 만들 수 있으며 테이블이 행 수(레코드 수)에 대한 제한은 사용자가 지정하지 않는 한 없습니다.

테이블의 데이터(레코드) 입력

테이블에 데이터를 넣고 빼는 작업은 IDLdbRecordset 객체가 담당합니다. 이는 앞에서 생성한 IDLdbDatabase의 객체(oDB)를 통해 연결됩니다.

```
IDL> oRS=obj_new('IDLdbRecordset', oDB, table='planets')
IDL> oRS->AddRecord, 'Mercury', 0.3871, 0.24085, $
                     0.0553, 0
IDL> oRS->AddRecord, 'Venus', 0.7233, 0.61521, $
                     0.8149, 0
.... 실제 프로그램에서는 반복문을 통해 데이터를 입력합니다 ...
```

이 예제에서는 문자열, 정수, 실수 등 평범한 데이터 타입만을 다루지만, 바이너리(이미지), 날짜 등 프로그래머가 다루는 거의 모든 데이터 타입을 정의하여 사용할 수 있습니다. 'Mercury' 행성을 또 넣으려 시도해 봅니다.

```
IDL> oRS->AddRecord, 'Mercury', 0.3871, 0.24085, $
                     0.0553, 0
%IDLDBRECORDSET::ADDRECORD:  ODBC  [MySQL].....
Duplicate entry 'Mercury' for key 1
```

하나의 테이블에 Mercury가 두 개 이상 들어 가는 것은 우리가 원하던 바가 아닙니다(테이블 생성에서 name 필드는 primary key로 설정되어 중복되면 안되는 조건을 걸었습니다). 이와 같이 데이터베이스 운영 중에 원치 않는 데이터가 입력되는 것을 사전에 방지하는 것도 데이터베이스 사용의 큰 장점입니다.

데이터베이스 연결 끊기

```
IDL> obj_destroy, oDB
```

데이터베이스 연결 객체를 소멸시키면 연관되어 있는 IDLdbRecordset 객체도 함께 소멸되며 데이터베이스와의 연결도 종료됩니다.

데이터베이스에서 데이터 읽어오기

```
;DB 연결
oDB=obj_new('IDLdbDatabase')
oDB->Connect, datasource=.., user_id=.., password=..
```

테이블에 있는 모든 행을 읽어오는 방법은 다음과 같습니다. 한 행(레코드)은 IDL에서 구조체로 처리됩니다.

```
oRS=obj_new('IDLdbRecordset', oDB, table='planets')
ok=oRS->MoveCursor(/FIRST)
one_rec=oRS->getRecord()
help, one_rec, /struct
while oRS->moveCursor(/NEXT) do begin
        one_rec=oRS->getRecord()
        help, one_rec, /struct
endwhile
```

SQL문의 Select 문을 이용하여 다양한 형태의 데이터 조회가 가능합니다. 데이터베이스 사용의 장점 중 하나는 다양한 조회 방법이 이미 표준을 따라 모두 개발 되어 있다는 것입니다.

다음 예는 행성 궤도반경이 1.0AU보다 큰 외행성을 조회하는 SQL 문의 사용 예입니다. getRecord()를 사용하는 방법은 위와 같습니다.

```
sql="select * from planets where dist > 1.0"
oRS=obj_new('IDLdbRecordset', oDB, SQL=sql)
```

다음 예는 행성의 질량이 큰 것부터 데이터를 읽어내는 방법입니다. desc를 asc로 고치면 질량이 작은 것부터 읽습니다.

```
sql="select * from planets order by mass desc"
oRS=obj_new('IDLdbRecordset', oDB, SQL=sql)
```

테이블의 데이터 수정/삭제/입력

IDLdbDatabase 객체는 SQL문으로 실행하는 모든 것을 수행할 수 있습니다. 수정/삭제/입력은 다음과 같은 표준 SQL 구문을 사용합니다. (앞에서 보았듯이, 데이터 입력은 IDLdbRecordset::AddRecord를 이용해도 됩니다).

수정	update 테이블명 set 필드명1=..., 필드명2=... [, where ...]
삭제	delete from 테이블명 where ...
삽입	insert into 테이블명 (필드명1, 필드명2, ...) values (필드값1, 필드값2, ...)

```
IDL> SQL="update into planets period=period*365.25"
IDL> oDB->executeSQL, SQL
```

위 예를 실행하면 테이블의 행성 공전주기를 지구 기준이 아닌 날짜 수로 변경합니다.