C++Builder ADO Programming (3) - ConnectionString Core

이전 강의에서 Connection 객체의 생성과 해제에 대해서 살펴 보았다. TADOConnection Component를 사용한다면 간단히 원하는 곳에 떨어뜨려 인스턴스를 생성하고 연결을 설정한 뒤 사용하고 연결을 끊을 수 있었다. 이 점에 있어선 Connection 객체의 해제는 신경을 쓸 필요가 없었다. TADOConnection Class를 사용하여 코드를 통해 Connection 객체를 동적으로 생성한다면 연결을 설정하고 사용한 후 연결을 닫은 뒤 Connection 객체를 명시적으로 해제 시켜야만 했다. 두 가지 방법 모두 어려운 점은 없었으며 간단한 것을 알 수 있었다.

자 그럼 이번에는 이전 강의에서 생성한 연결을 설정하는 방법에 대해 알아보자. 먼저 Connection 객체를 통해 어떤 데이터 공급자에 어떻게 연결할 것인지를 지정하기 위한 ConnectionString 속성부터 시작하자.

ConnectionString 속성

ConnectionString 속성은 Connection 객체로 어떤 데이터 원본에 연결하는 데 필요한 모든 정보를 담는다. 이전 강의에서 Open 메소드를 호출하여 Connection 객체를 열기 전엔 항상 이 속성이 미리 지정되어 있어야 한다. 이 속성에 포함될 수 있는 정보를 살펴 보자.

★ OLEDB를 위한 데이터 공급자(이는 Provider 속성을 통해서도 접근 가능하다)
★ 데이터 원본 (서버 이름, SQL 서버를 사용하는 경우)
★ 접근하고자 하는 기본 데이터베이스
★ 사용자 이름과 패스워드
★ ODBC 데이터 원본 이름 --- DSN을 사용하는 경우

데이터 공급자에 따라 위의 항목들을 모두 요구하기도 하고 일부만 요구하기도 하는데 그렇다면 이 속성에 포함되는 항목들을 구체적으로 알아보자.

항목	설명
Provider	데이터 공급자 - 데이터 원본을 위한 드라이버
Data Source 혹은 Server	데이터베이스의 서버 이름 또는 데이터 원본 파일 이름
Initial Catalog 혹은 Database	데이터베이스 이름
User Id 혹은 UID	데이터베이스에 연결하고자 하는 사용자의 이름.
Password 혹은 PWD	사용자의 패스워드
File Name	공급자의 연결 정보를 담고 있는 파일 이름
Remote Provider	RDS를 사용하는 경우 공급자의 이름
Remote Server	RDS를 사용하는 경우 서버의 이름
URL	데이터 원본의 역할을 할 파일 또는 폴더의 위치
DSN	ODBC를 사용할 경우 Data Source Name

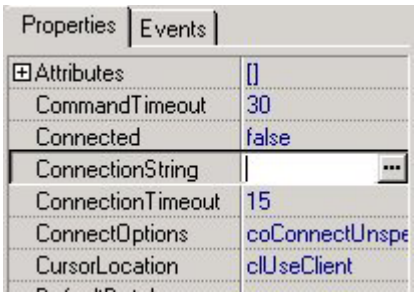
위의 표에서 File Name과 Provider는 상호 배타적이라는 점을 주의하기 바란다. Provider를 통해 ConnectionString에 공급자 이름을 포함시킨 경우 공급자 정보를 위한 파일 이름을 지정할 필요는 없으며 그 반대도 마찬가지이다. URL 항목 또한 그런데 Provider를 통해 공급자를 설정한 경우 URL 항목을 설정할 수 없다. URL 항목만을 지정했을 경우 데이터 공급자가 인터넷 게시를 위한 OLEDB 공급자 MSDAIPP.DSO로 자동적으로 설정된다.
Remote Provider와 Remote Server는 RDS와 함께 사용되는 것으로 서버의 공급자와 데이터 원본 정보를 뜻

한다. URL 항목은 ADO 2.5 버전에 새로 도입된 것으로 파일이나 디렉터리에 대한 연결을 여는데 쓰인다. 파일이나 디렉터리를 연 후에는 파일들과 폴더들을 관리할 수 가 있다. 이 두 가지 사항에 대해서는 이후의 강의에서 살펴보도록 하자.

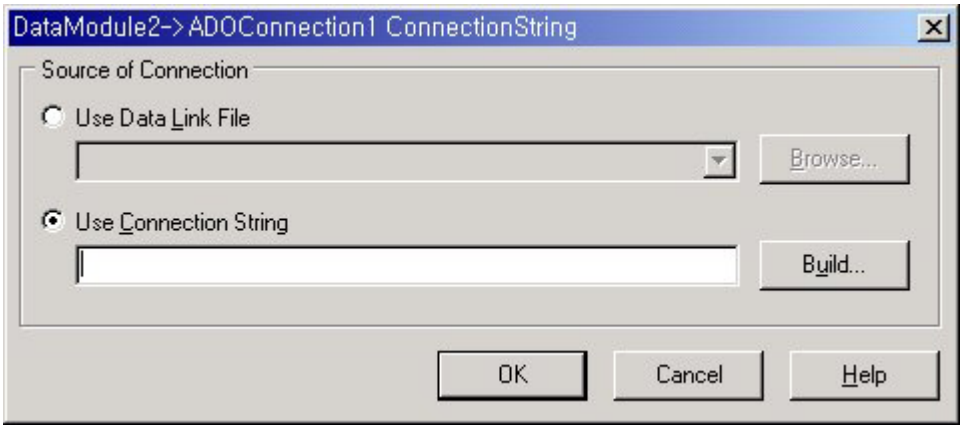
ConnectionString 속성은 위의 항목들의 조합으로 구성되며 두 가지 방법으로 설정할 수 있다. 첫번째는 Object Inspector를 통한 것으로 ConnectionString 속성창의 … 부분을 클릭하면 ConnectionString 생성 창이 뜨며 자신에 맞는 설정을 차례차례 입력하면 ConnectionString이 자동적으로 만들어진다. 두 번째 방법은 자신이 직접 위의 항목들을 조합해 코딩을 통해 ConnectionString을 생성하고 설정하는 방법이 있다. 이 방법은 코드를 통해 동적으로 TADOConnection의 인스턴스를 생성했을 때 ConnectionString을 설정하는 방법이며 Visual 하지 않은 어플리케이션을 제작할 때 유용한 방법이다. 그렇다면 이 두 가지 방법에 대해 자세하게 알아보도록 하자.

먼저 첫번째 방법을 살펴보자. 예제로 우리의 레밍은 다음의 작업을 통해서 생성된 ConnectionString 설정으로 MS SQL Server에 접속할 것이다.

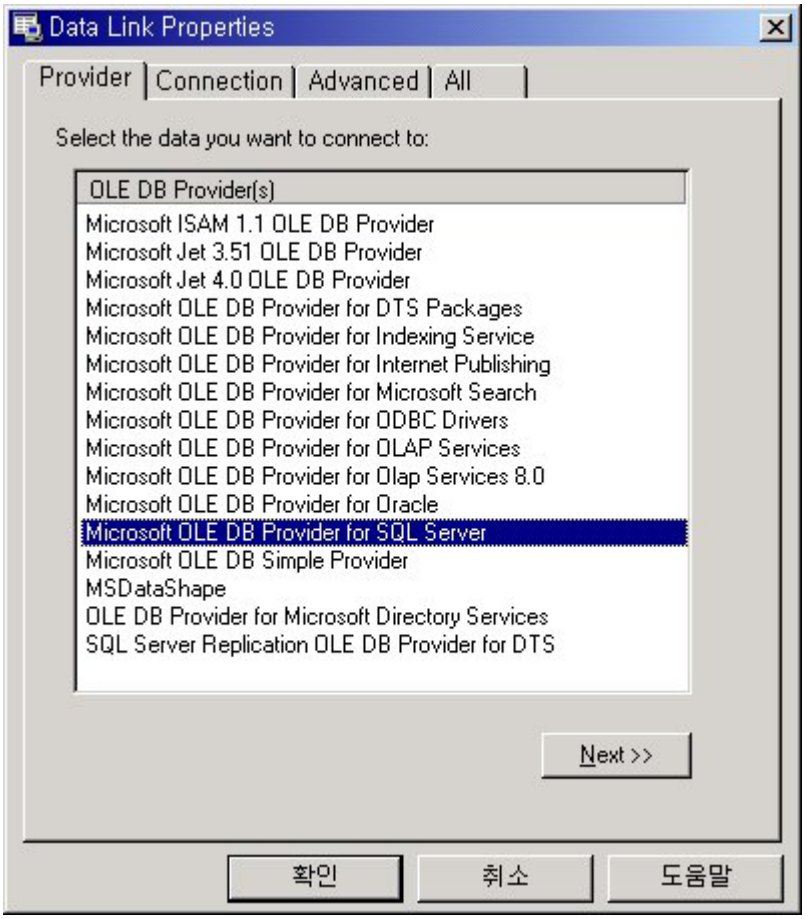
1. ADOConnection Component를 클릭한 후 Object Inspector의 ConnectionString 속성 부분의 … 버튼을 클릭한다.



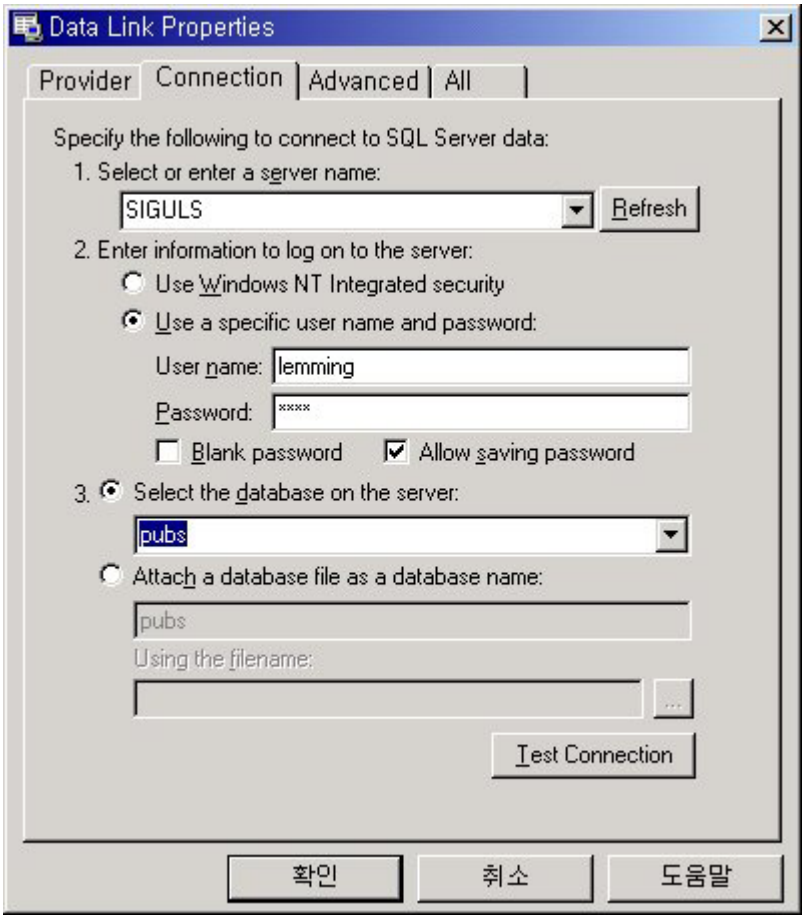
2. 갈림길이 나왔다. 두 번째 항목이 Default로 지정되어 있다. 첫번째 use Data Link File 은 나중에 알아보도록 하자. Build 버튼을 클릭한다.



3. 앞의 표에서 설명한 Provider 항목을 설정할 수 있는 창이 뜬다. 자신에게 적합한 Provider를 설정하고 Next 버튼을 누른다. 함 살펴보면 여러 가지 종류의 Provider들이 있는 것을 볼 수 있다. 이 리스트들은 로컬 머신의 셋팅 상태에 따라 다르다. 우리의 레밍은 SQL Server용 OLEDB Provider를 선택했다.



4. Provider를 선택한 후 Next 버튼을 누르면 나머지 세부 항목을 설정하는 Connection 탭으로 전환된다. 정보를 입력하자. 예제에서는 서버 이름이 SIGULS, 데이터 원본인 SQL Server의 사용자 명이 lemming, 패스워드는 0000이고 MS-SQL Server의 기본 예제 데이터베이스인 pubs에 SQL Server 인증으로 접속한다.



이것만으로도 ConnectionString을 생성하는데 필요한 핵심 사항들은 충분하므로 위의 모든 입력을 마쳤다면 Test Connection 버튼을 클릭하자. 만약 연결과 정보에 문제가 없다면 다음과 같은 Modal 창이 뜬다.

Test Connection이 성공했다면 확인 버튼을 누르고 위의 데이터 링크 프로퍼티 창을 닫으면 ConnectionString이 만들어 진다. 위의 작업에서 우리의 레밍이 만든 ConnectionString을 한 번 살펴보자.

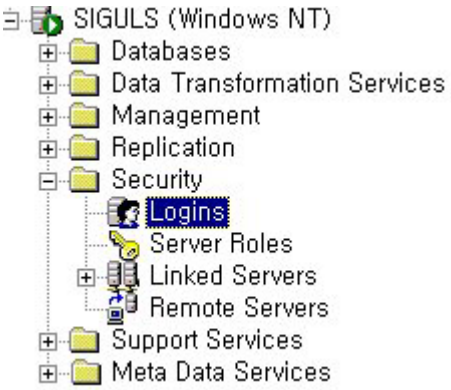
Provider=SQLOLEDB.1;Data Source=SIGULS;Password=0000;User ID=lemming;
Initial Catalog=pubs;Persist Security Info=True

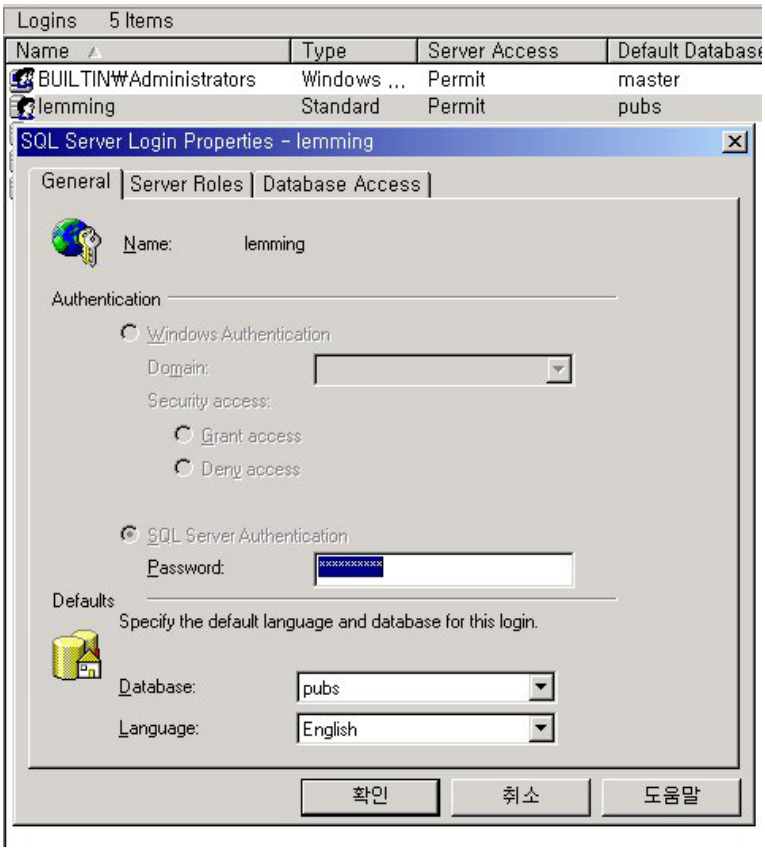
빨간 글씨의 ConnectionString, 위의 표의 항목들, 연결 정보를 다시 한 번 살펴보자.

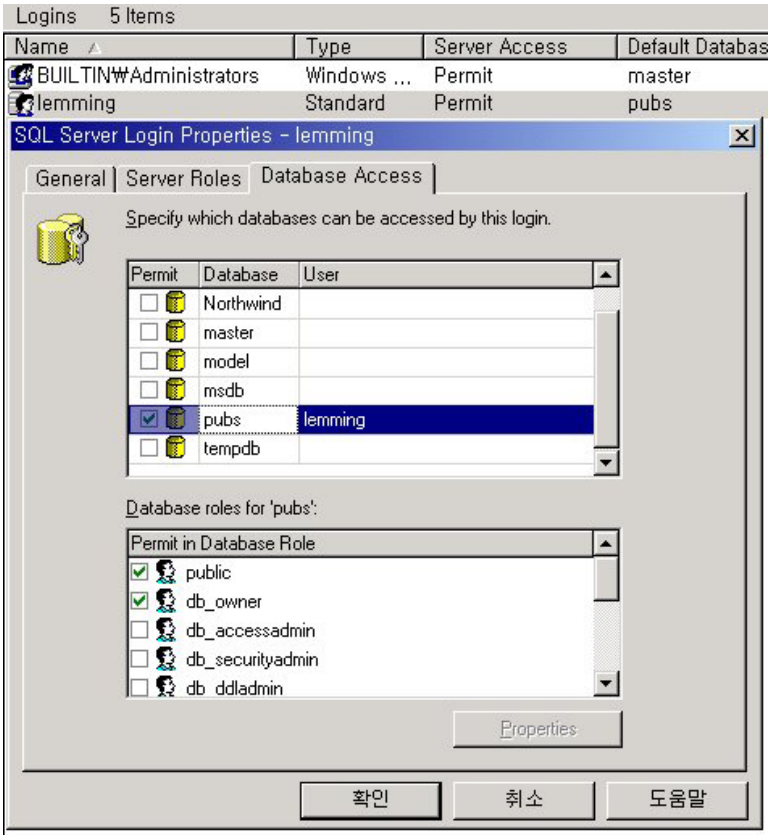
Provider=SQLOLEDB.1;
데이터 공급자가 MS SQL Server 용 OLEDB 공급자라는 것을 의미한다.

Data Source=SIGULS;
데이터베이스 서버 (인스턴스) 네임이 SIGULS라는 것을 나타낸다. Server=SIGULS;을 사용해도 된다. 여기서 클라이언트 어플리케이션과 데이터베이스 서버가 다른 머신에 있는 --- 서버가 지역 서버가 아닌 --- C/S 어플리케이션이나 웹 서버와 데이터베이스 서버가 다른 머신에 각각 분리되어 있는 웹 어플리케이션, 그리고 비즈니스 Component 층(흔히 말하는 MTS / COM+ 층, 일종의 어플리케이션 서버 층)과 데이터베이스 서버가 역시 다른 머신에 분리되어 있다면 Client Network Utility로 생성한 Alias명을 써야 한다. 일단은 그렇게 알아두라. 그것은 또 하나의 주제로서 MS-SQL을 개발자의 눈에서 다룬 차후의 장에서 논의하도록 하자.

Password=0000;User ID=lemming;
뭐 이것은 직관적으로 알 수 있을 것이다. SQL Server의 인증을 받은 lemming이라는 사용자(로그인 명)가 있고 그것의 패스워드가 0000이라는 말이다. 당연히 이것은 SQL Server 내의 설정 정보로 존재해야 됨을 의미한다. 아래의 그림은 SQL Server에 설정되어 있는 레밍에 대한 로그인 정보이다.







Persist Security Info=True;

이것은 SQL Server 인증을 가지고 위에서 설명한 로그인 정보로 데이터베이스 서버에 연결을 얻겠다는 의미이며 바로 위 줄의 로그인 정보가 ConnectionString내에서 사용되면 자동적으로 설정되는 사항이며 생략 가능하다. 만약 잠재적인 데이터베이스 보안의 위험성을 경계하거나 국지적이고 폐쇄적인 소규모의 Windows

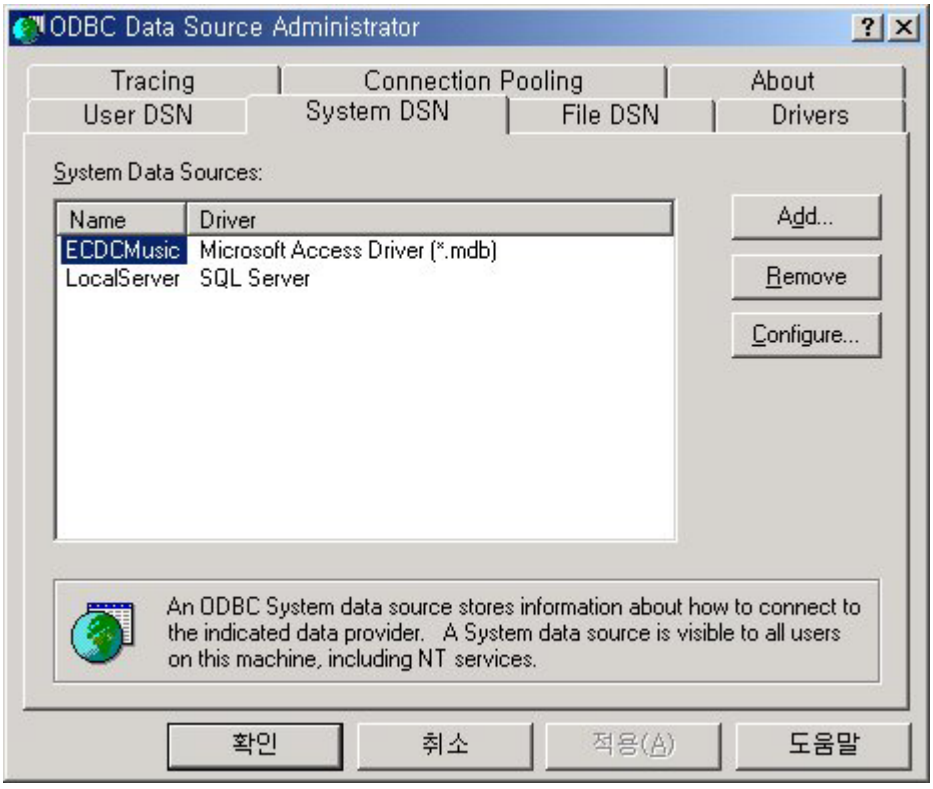
Network로 구성되어 있으며 클라이언트 웍스테이션으로 로그인하는 LAN 상의 사용자만을 위한 어플리케이션을 작성시에는 SQL Server 인증을 사용하는 대신 Windows NT 통합 인증을 사용하기도 하는데 --- 물론 여기서 전제되어야 하는 것은 훌륭한 도메인 모델이나 Active Directory, 그룹과 계정 정책 등이 하부구조로 잘 구축되어 있어야 된다 --- 그때의 ConnectionString은 이 부분이 다음과 같이 바뀐다.
Integrated Security=SSPI;Persist Security Info=False;

Initial Catalog=pubs;
위의 사용자 정보로 연결을 해서 사용할 수 있는 기본적인 데이터베이스 이름이다. 이 속성은 MS SQL Server가 데이터베이스 위주의 스키마를 가진 RDBMS이기 때문에 설정 가능한 항목이며 Database=pubs;로 해도 된다. 덧붙여 설명하자면 이 속성은 Oracle을 RDBMS로 사용하는 경우엔 설정되지도 않고 쓰이지도 않는다. Oracle은 기본적으로 사용자 위주의 스키마를 가진 RDBMS이기 때문이다. Oracle을 사용할 때의 ConnectionString을 설정하는 방법에 대해서는 차후의 강의에서 살펴보도록 하자.

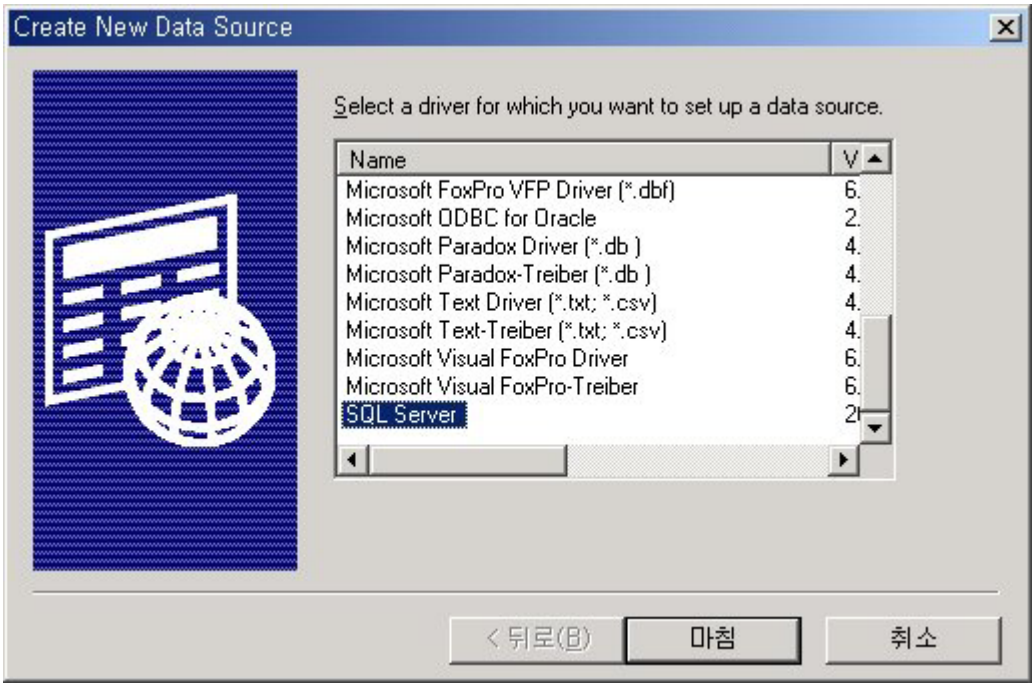
자 이것으로 ADO를 통해 OLEDB를 사용하는 MS SQL Server 용의 ConnectionString이 다 만들어졌다. 그러면 이제 다른 방법을 한 번 알아보자. 분명 이 서툰 강의를 그나마 유심하게 본 사람이라면 첫번째 강의에서 필자가 그린 어플리케이션과 데이터 저장소 사이의 여러 층에 대한 그림을 기억할 것이다. 거기를 보면 맨 위의 같은 ADO층을 거치더라도 직접 자체 Native OLEDB Provider를 통한 방법이 있고 ODBC에 대한 OLEDB Provider들을 통한 방법이 있다는 것을 알 수 있었을 것이다. 혹시 그림이 서투르고 형편 없어서 몰랐다면 이번 기회에 그 그림이 이 말이었다는 것을 알아두자. 그리고 이번 강의 바로 앞의 예제에서 우리의 레밍은 Native OLEDB Provider for SQL Server를 사용한 ConnectionString을 얻을 수 있었다. 알아볼 다른 방법이란 ODBC에 대한 OLEDB Provider들을 사용한 ConnectionString을 얻는 방법이다.
그렇다면 왜 연결 층을 한 층 더 거치고 속도도 느린 방법을 알아보려고 하는지에 대한 의문을 가질 것이다. 그러나 분명히 이 문제는 성능을 떠난 신뢰성의 문제와 밀접한 관련이 있다. 사실 간략하게 말하면 ODBC는 표준에다가 광범위하게 지원 받고 있고 널리 사용되고 있는데 OLEDB는 표준이 아니기 때문이다. 또 지나치게 MS 지향적이라 항상 신뢰성을 기대할 수 있는 만능이 아니기 때문이다. 이것은 ORACLE이나 SYBASE같은 다른 RDBMS를 사용할 때 연결의 문제에서 절실하게 느끼게 되는 어려움 중의 하나인데 아무튼 이 방법을 알아두면 나름대로 해결책이 하나 더 생기므로 유용하다.

또 한가지 레밍이 그린 서툰 그림에서 ADO를 거치지 않고 직접 OLEDB를 사용하여 연결하는 방법을 볼 수가 있는데 그것이 속도도 제일 빠르지만 생산성에 대해서는 의문을 가질만하며 분명히 삽질이고 이 강의에서 논외의 문제라 다루지 않겠다.
자 레밍을 재촉해 서둘러 가자. 이전 예제와 같은 설정으로 MS SQL Server에 접속하자.

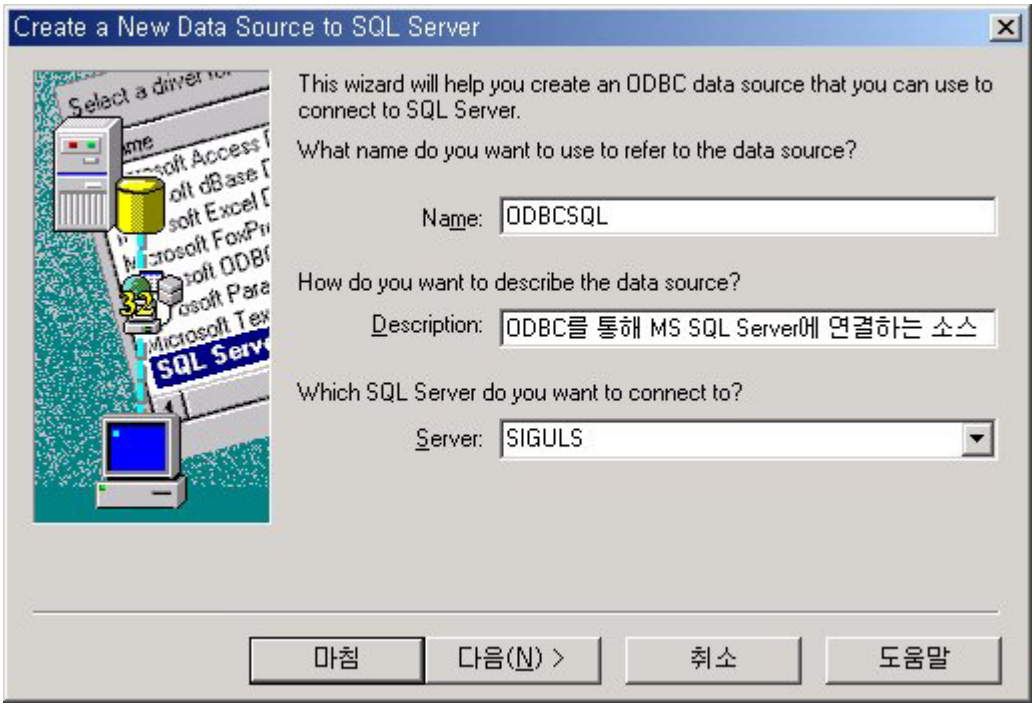
1. 우리의 레밍은 ODBC Driver들을 위한 OLEDB Provider를 통해 연결을 얻어야 하므로 먼저 MS SQL Server에 접속할 수 있는 ODBC DSN(데이터 소스 네임)이 있어야 한다. 없으므로 하나 만들자. ODBC Data Source Administrator 창을 띄우고 System DSN 탭을 클릭하면 현재 이 로컬 머신에 있는 ODBC 데이터 원본들의 목록을 볼 수 있다. 하나 만들기로 했으니깐 Add 버튼을 누르자.



2. 버튼을 누르면 Create New Data Source 창이 뜨면서 각종 드라이버들의 목록이 보인다. 원하는 것을 택하고 마침 버튼을 누르면 되는데 우리의 레밍은 MS SQL Server를 택하고 그 작은 발로 마침 버튼을 지긋이 눌렀다.

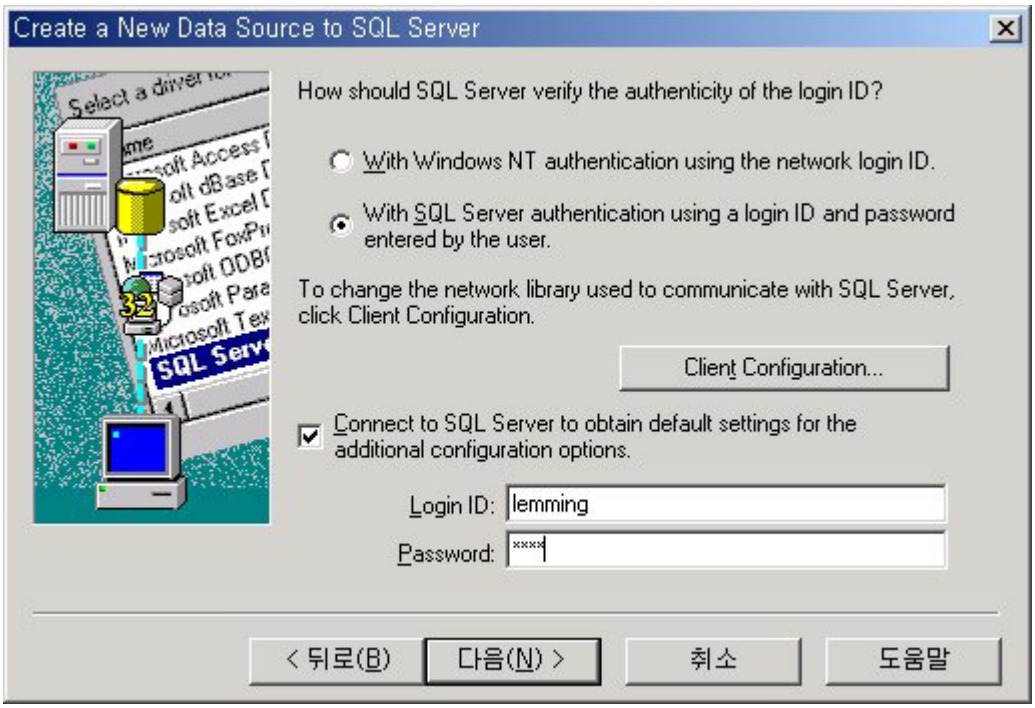


3. SQL Server에 대한 새로운 데이터 소스 네임을 설정하는 부분이다. 이름과 설명, 그리고 서버를 입력한다. 원하는 이름을 입력하고 서버를 선택한다. 앞의 예제와 같은 설정이므로 서버 명은 같은 SIGULS이다.

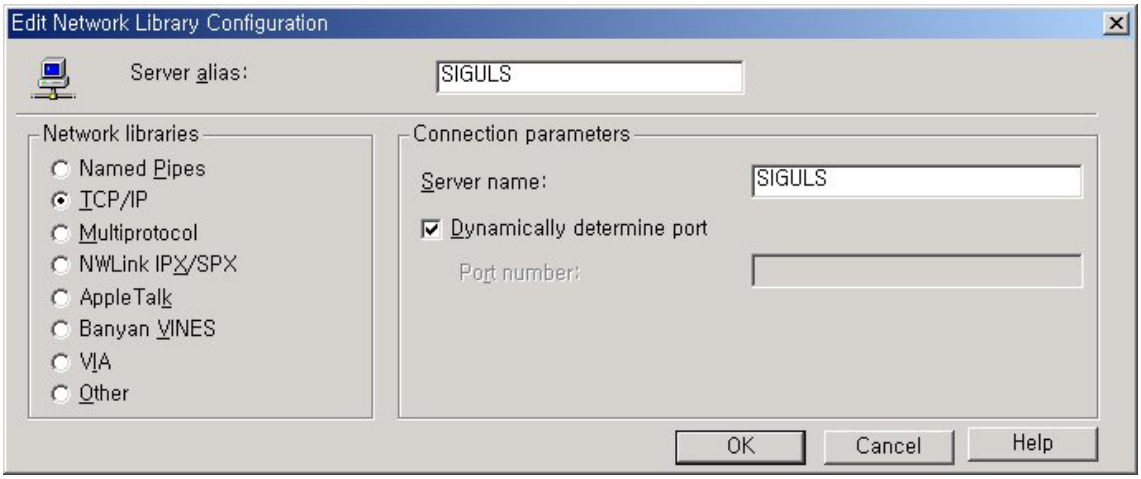


서버 ComboBox를 선택하면 현재 이 컴퓨터에서 연결할 수 있는 SQL Server의 목록들이 표시된다. 물론 Client Network Utility로 설정한, 외부 서버에 접속할 수 있는 Alias 목록들도 표시된다. 정보를 입력한 후 다음 버튼을 누른다.

4. 다음은 새로운 SQL Server에 대한 Data Source의 로그인 정보와 인증 방법을 설정하는 단계이다. 아래의 그림에서 설정한 것과 같이 우리의 레밍은 이전의 정보대로 SQL Server 인증과 lemming 사용자 정보를 가지고 정보를 입력했다.

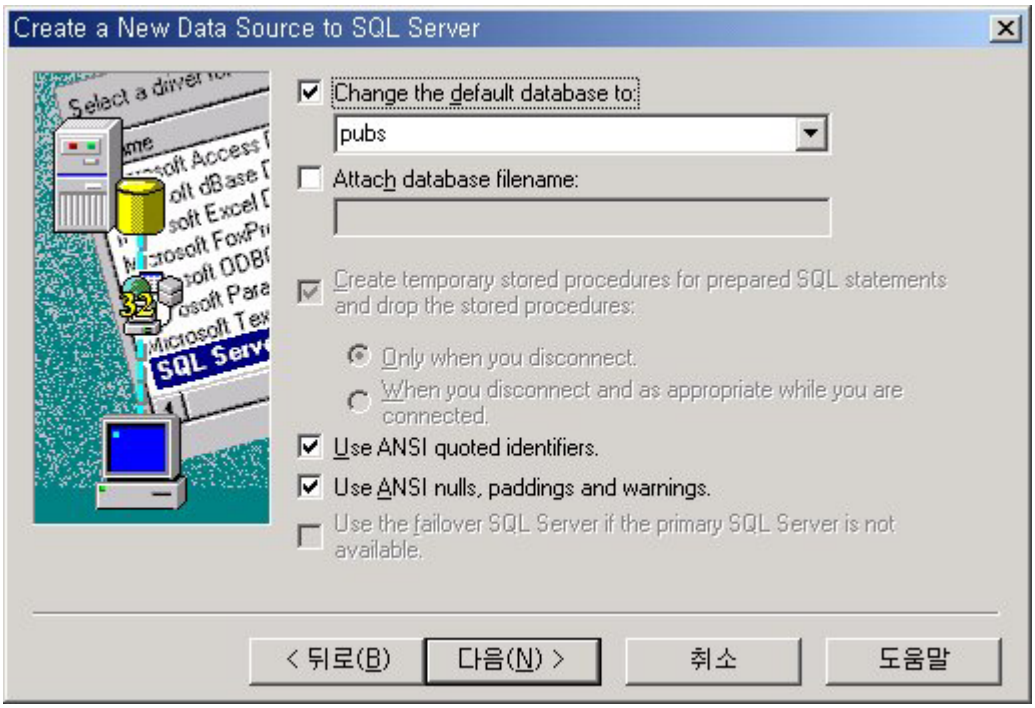


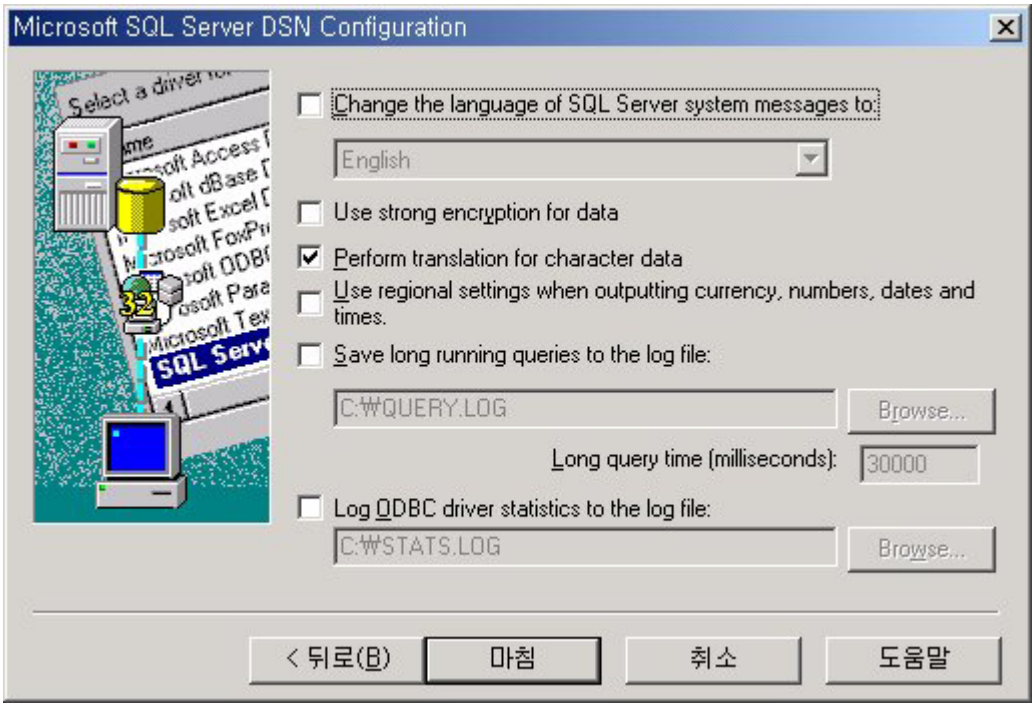
Client Configuration 버튼을 눌러 SQL Server와 통신하는데 사용되는 네트웍 라이브러리를 바꿀 수 있다. 한번 눌러보면 아래와 같은 창이 뜨며 왼쪽에서 네트웍 라이브러리를 바꿀 수 있다.



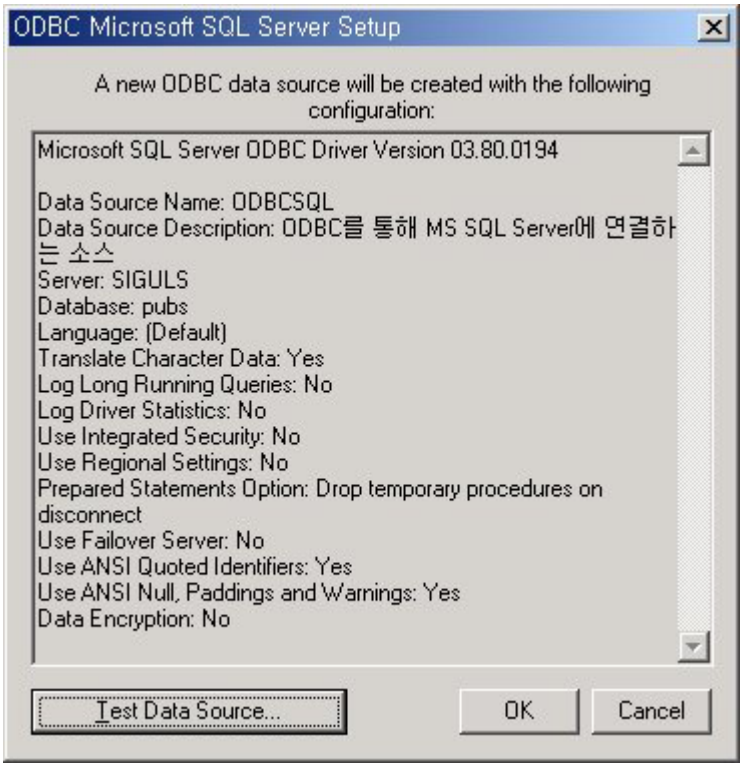
그러나 일반적으로 TCP/IP가 무난하며 다른 네트웍 라이브러리와 이와 관련된 자세한 사항에 대해서는 별도의 강의를 통해 알아보자. 여기서 중요한 것은 Server Alias와 Connection Parameters 부분의 Server name 부분이다. Server Alias는 다른 이름으로 바꿔도 상관 없지만 복잡하니 그냥 두기로 하자. 만약 원격의 데이터베이스 서버에 접속하려 한다면 데이터베이스 서버의 IP 주소를 Server name에 입력한다. 여기서는 로컬 데이터베이스 서버이므로 디폴트 설정을 그대로 둔다. Ok 버튼을 누르고 원래 창으로 돌아와 다음 버튼을 누르자.

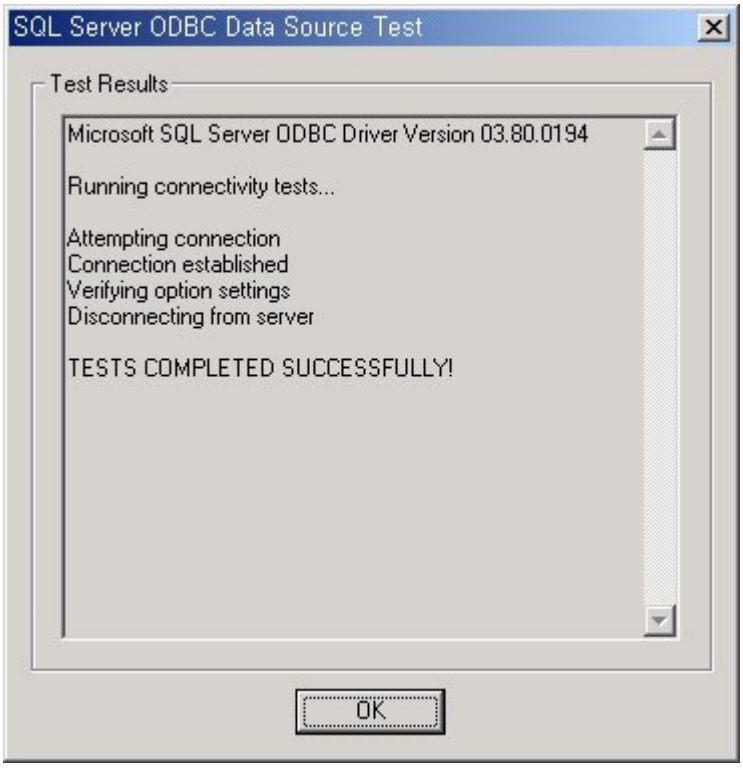
5. 다음은 기본 데이터베이스 설정, 준비된 SQL 문에 대한 처리방식과 각종 ANSI 규칙들과 언어 및 데이터 타입에 대한 세부 설정, 로그와 로그 파일에 대한 각종 세부 설정을 마무리하는 작업이다.





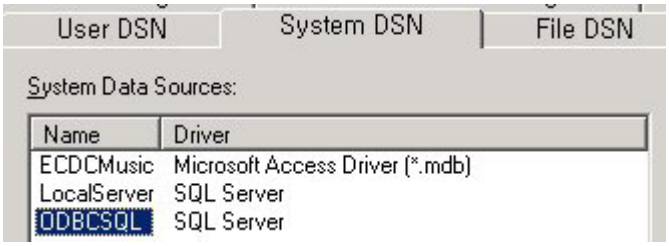
6. 설정을 다하고 마침 버튼을 누르면 지금까지 설정한 정보들의 리스트와 Data Source를 테스트 하는 창이 뜬다. Test Data Source 버튼을 눌러서 테스트를 해보자.



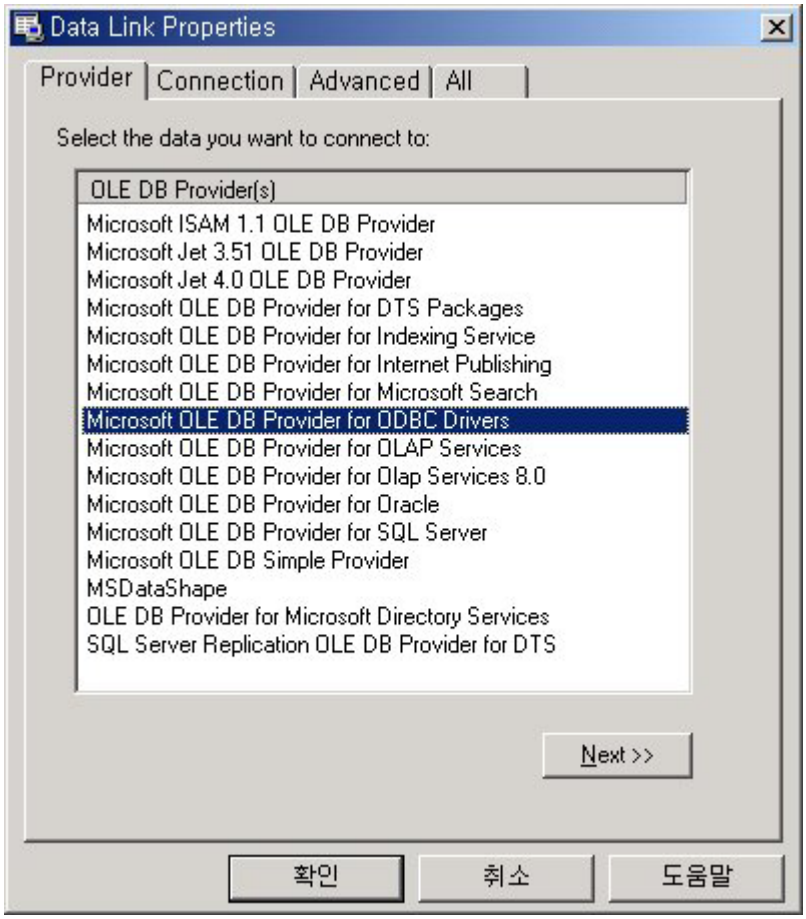


테스트가 성공했다는 창이 뜨면 Data Source에 대한 설정이 올바로 되었다는 것이며 만약 성공하지 못했다는 메시지가 뜨면 창을 닫고 이전의 단계로 돌아가서 여러 연결 정보들을 다시 한번 살펴보고 올바른 설정을 해 다시 연결을 테스트 하길 바란다. 연결에 대한 문제는 데이터베이스 프로그래밍을 하는데 있어서 전제가 되는 작업이며 때때로 많은 시간이 드는 작업이다. 천천히 그리고 꼼꼼하게 여러 정보를 살피고 올바른 설정을 하자.

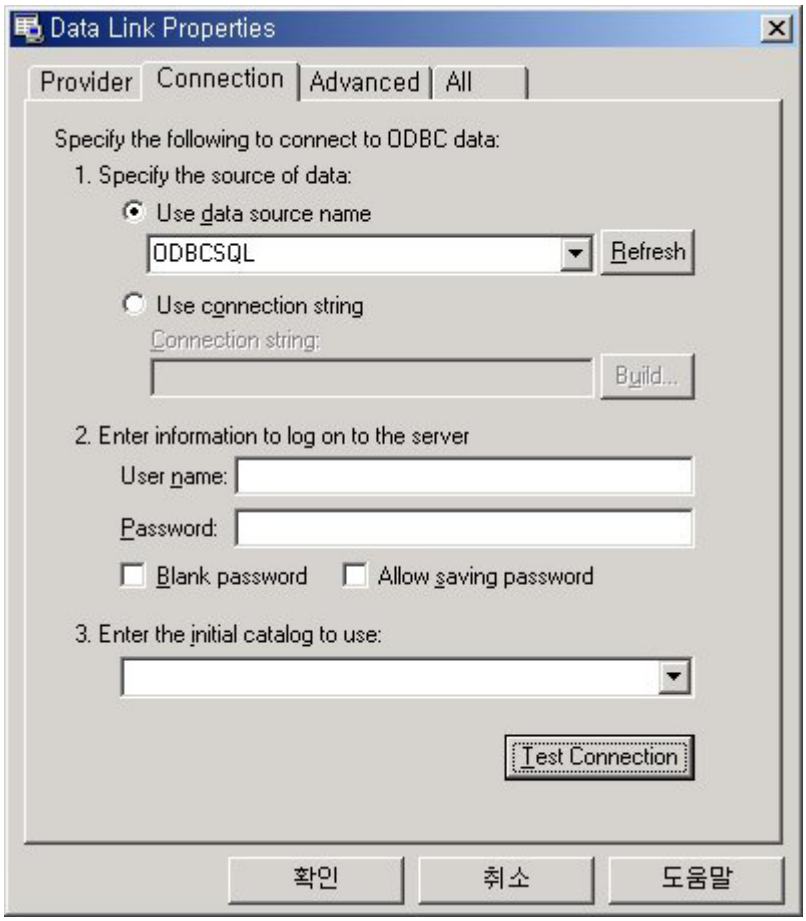
7. ODBC Data Source에 대한 설정을 마치고 연결도 성공하였으므로 모든 창을 닫자. 그러면 ODBC Data Source Administrator 창의 System DSN 탭에 우리의 레밍이 만든 ODBC Data Source for SQL Server가 보인다.



8. ODBC Data Source for SQL Server가 만들어졌으므로 다시 TADOConnection Component로 돌아가 ConnectionString을 만들자. 이전 예제 3의 과정으로 돌아가 OLEDB Provider 들의 리스트에서 ODBC Driver 들을 위한 OLEDB Provider를 선택한다.



9. Provider를 지정하고 Next 버튼을 눌러 연결에 대한 정보를 완성하자.



1번의 Use Data Source Name를 체크하고 미리 만들어둔 Data Source Name인 ODBCSQL을 사용하자. 단지 이것만을 설정해도 된다. 왜냐하면 ODBC 설정 정보에서 이미 모든 정보를 입력했기 때문이다. Test Connection 버튼을 눌러 연결을 검사하자.

역시 다음과 같은 창이 뜨면 연결에 성공한 것이다.

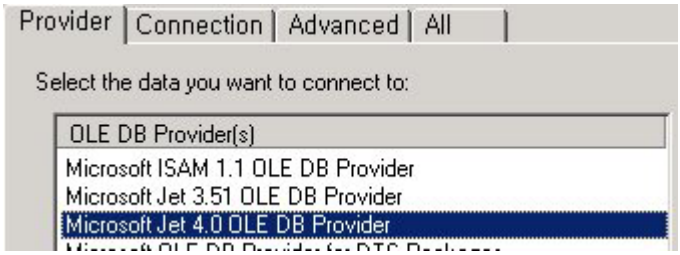
연결에 성공했다면 확인 버튼을 눌러 모든 사실을 확인한 뒤 ConnectionString을 얻자.
Provider=MSDASQL.1;Persist Security Info=False;Data Source=ODBCSQL

이번에는 위와 같은 ConnectionString을 얻었다. 유심히 볼만한 것은 Provider 속성으로 지정된 MSDASQL인데 이는 Object Inspector에서 TADOConnection Component의 Provider 프로퍼티 값으로 살펴볼 수 있다. 액세스 데이터베이스를 같은 방법 --- OLEDB for ODBC Driver for Access --- 으로 설정, 연결 하는 데에도 이 공급자가 사용된다. 만약 System DSN에 ODBCACCESS란 Access Driver 용 Data Source Name이 있다면 그에 대한 ConnectionString은
Provider=MSDASQL.1;Persist Security Info=False;Data Source=ODBCACCESS
가 된다. 이외에도 Provider 프로퍼티 값은 여러 가지가 있는데 이후의 강의에서 차례차례 소개될 것이다

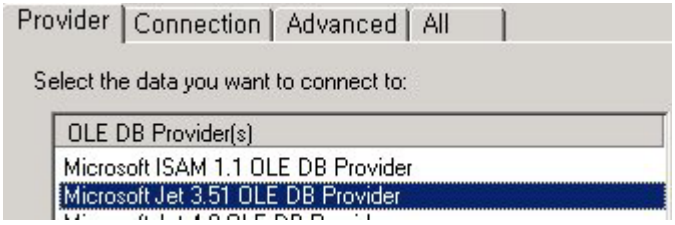
이것으로 MS SQL Server를 예로 native OLEDB Provider를 사용하여 ConnectionString을 생성하는 방법과 ODBC용 OLEDB Provider를 사용하여 ConnectionString을 생성하는 방법에 대해 자세하게 알아보았다. 훌륭한 로컬 데이터베이스인 Access의 경우도 기본적인 과정은 MS SQL Server와 비슷하니 중요한 과정과 결과만을 보자.

Access의 경우도 SQL Server와 같이 2가지 Provider와 방법을 통해 ConnectionString을 생성할 수 있다. 먼저 Access용 native OLEDB Provider로 ConnectionString을 얻자.

Access 2000의 경우 native OLEDB Provider는 아래의 Provider를 선택한다.

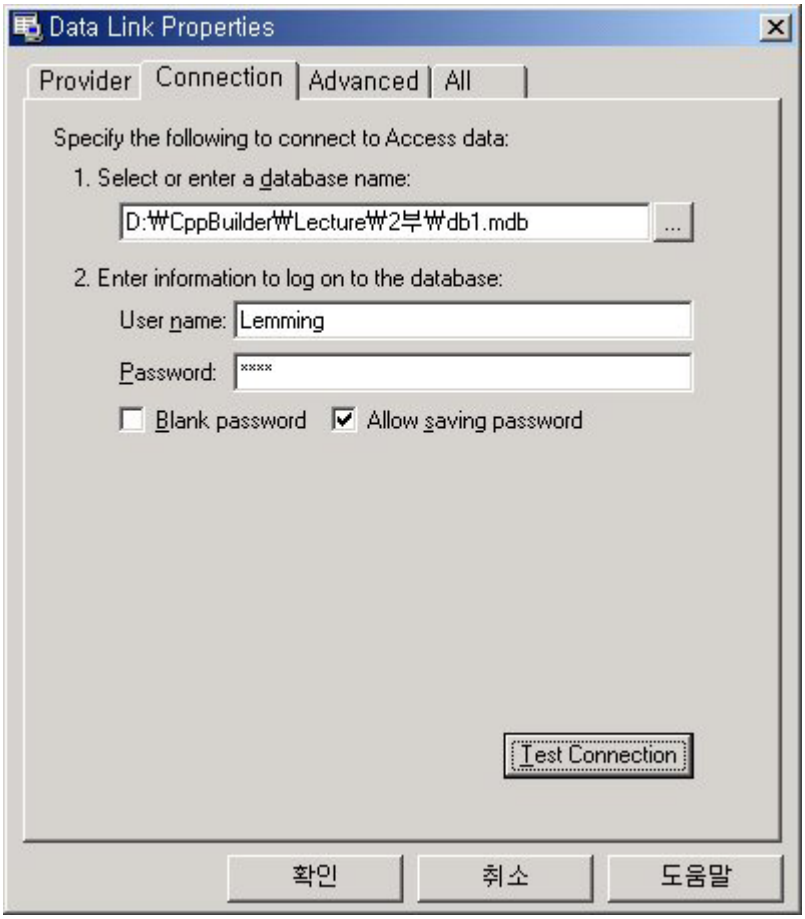


Access 97의 경우 native OLEDB Provider는 아래의 Provider를 선택한다.



위의 그림에서 Provider를 택하고 Next 버튼을 누르면 MS SQL Server와 같이 연결 정보를 설정하는 창이 뜬

다. 여기서 … 버튼을 클릭하여 데이터베이스 명을 입력하고 데이터베이스 Logon에 필요한 정보를 입력하자. 역시 사용자 정보는 SQL Server 때와 같다.



로그인 정보가 없는 경우나 필요하지 않은 경우 기입하지 않아도 된다. 하지만 설정은 같이 해보자. 역시 입력을 마친 후 Test Connection 버튼을 클릭하여 제대로 연결이 되는지 테스트하자.

역시 위의 창이 뜨면 연결에 성공한 것이다.

다음은 그렇게 하여 얻어진 ConnectionString이다. 먼저 Access 2000의 경우

Provider=Microsoft.Jet.OLEDB.4.0;Password=0000;User ID=Lemming;
Data Source=D:₩CppBuilder₩Lecture₩2부₩db1.mdb;Persist Security Info=True

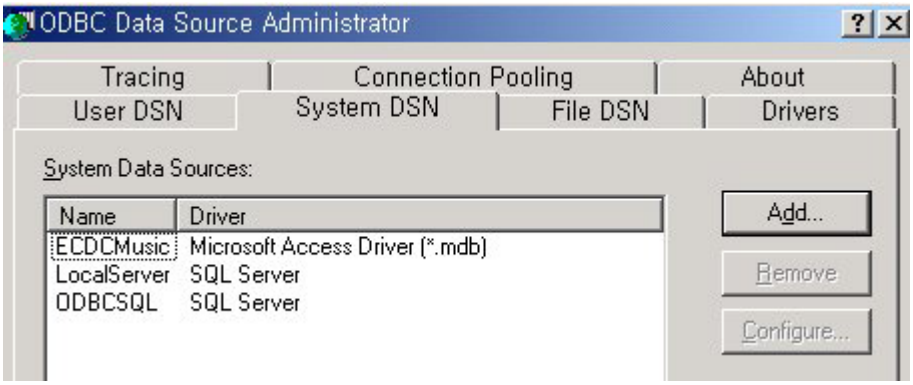
다음은 Access 97의 경우이다.

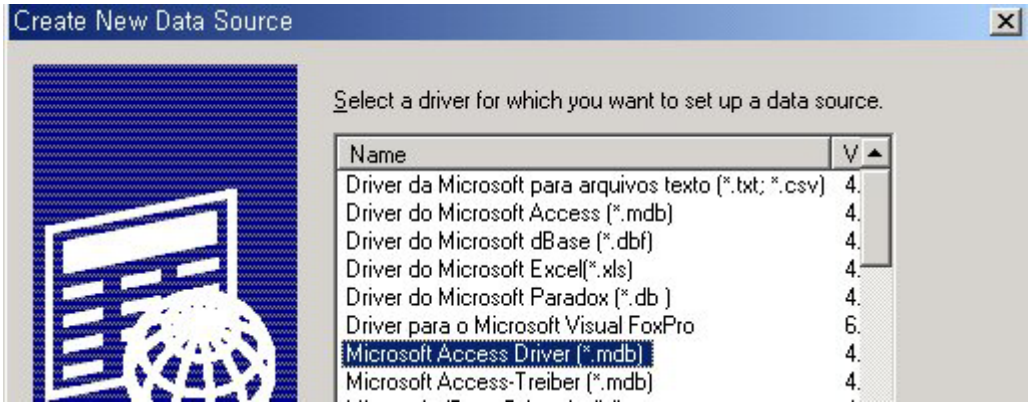
Provider=Microsoft.Jet.OLEDB.3.51;Password=0000;User ID=Lemming;
Data Source=D:₩CppBuilder₩Lecture₩2부₩db1.mdb;Persist Security Info=True

단지 Provider 항목의 값이 Access 용 native OLEDB Provider로 바뀌고 Data Source 항목의 값이 데이터베이스 서버 인스턴스 명에서 Access Database File의 패스로 바뀐 것을 제외하고는 SQL Server 때와 같다. 만약 Access 파일 자체에 사용자 아이디와 암호가 걸려있다면 위와 같은 ConnectionString들이 생성될 것이다. 그런 로그인 정보가 없다면 Password=0000;User ID=Lemming; 부분은 없어지고 Persist Security

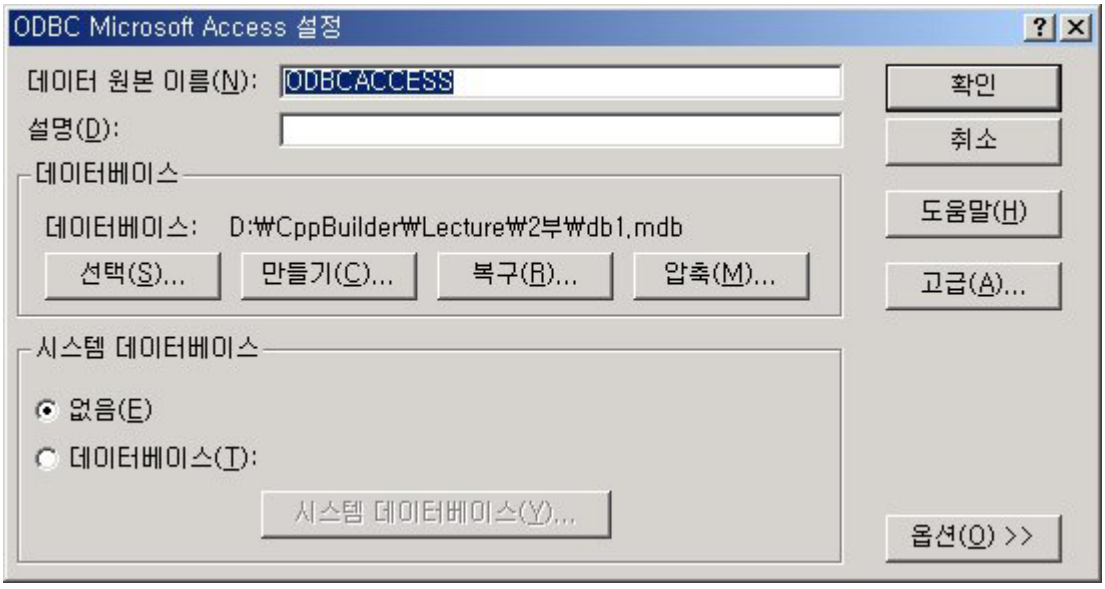
Info=False가 될 것이다. 물론 없어도 상관없다.

자 다음은 Access ODBC용 OLEDB Provider를 이용하는 방법이다. 먼저 Access용 ODBC Driver를 하나 만든다. 만드는 방법 역시 SQL Server 때와 비슷하다. ODBC Data Source Administrator 창을 띄우고 System DSN 탭에서 Add 버튼을 클릭하여 Microsoft Access Driver를 선택한다.



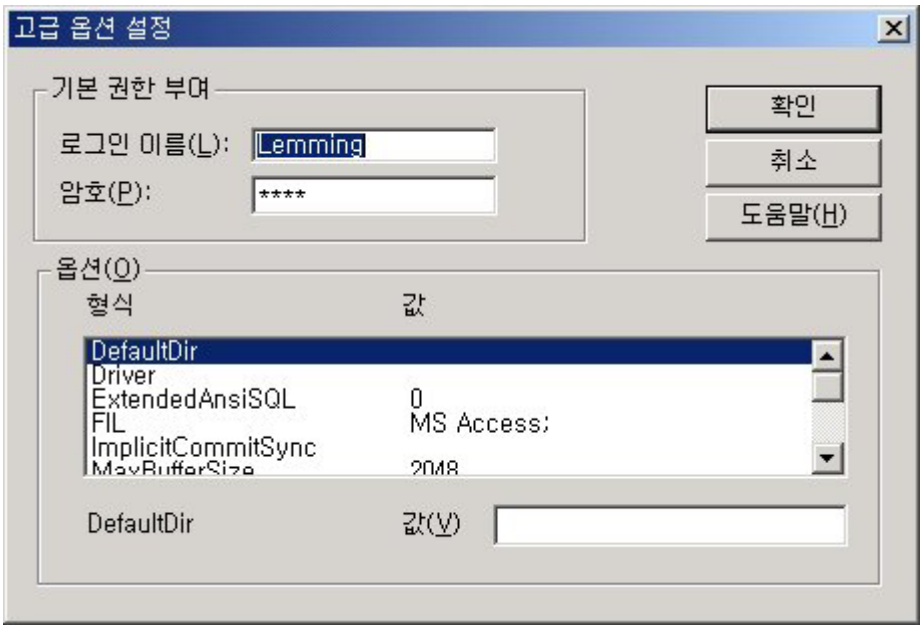


마침 버튼을 누르면 아래와 같은 ODBC Microsoft Access 설정 창이 뜬다 정보를 입력하자. 데이터 원본 이름에 원하는 이름을 써넣고 데이터베이스 선택 버튼을 눌러 액세스 파일을 선택한다.

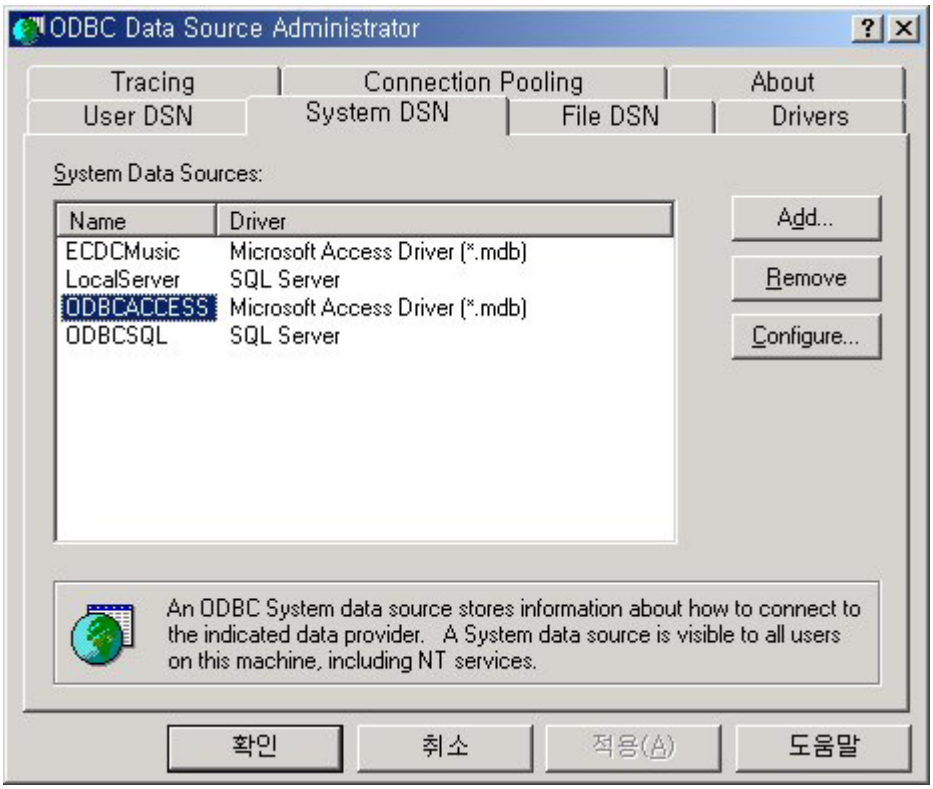


전의 예제에서 사용자 정보가 있었으므로 고급 버튼을 눌러 로그인에 대한 정보를 입력하자. 물론 액세스 파일에 대한 로그인 정보가 없다면 이 과정은 필요 없다. 나머지 옵션 항목도 정보가 필요하다면 설정하도록

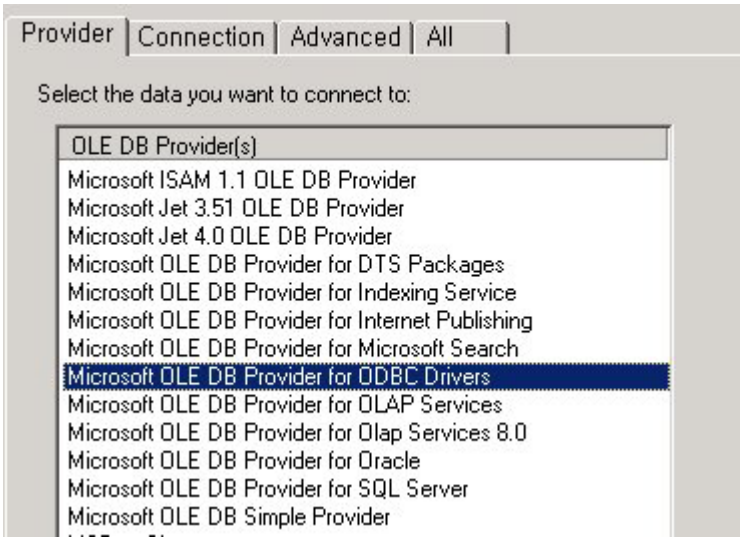
하자. 설정을 마쳤으면 확인 버튼을 누른다.



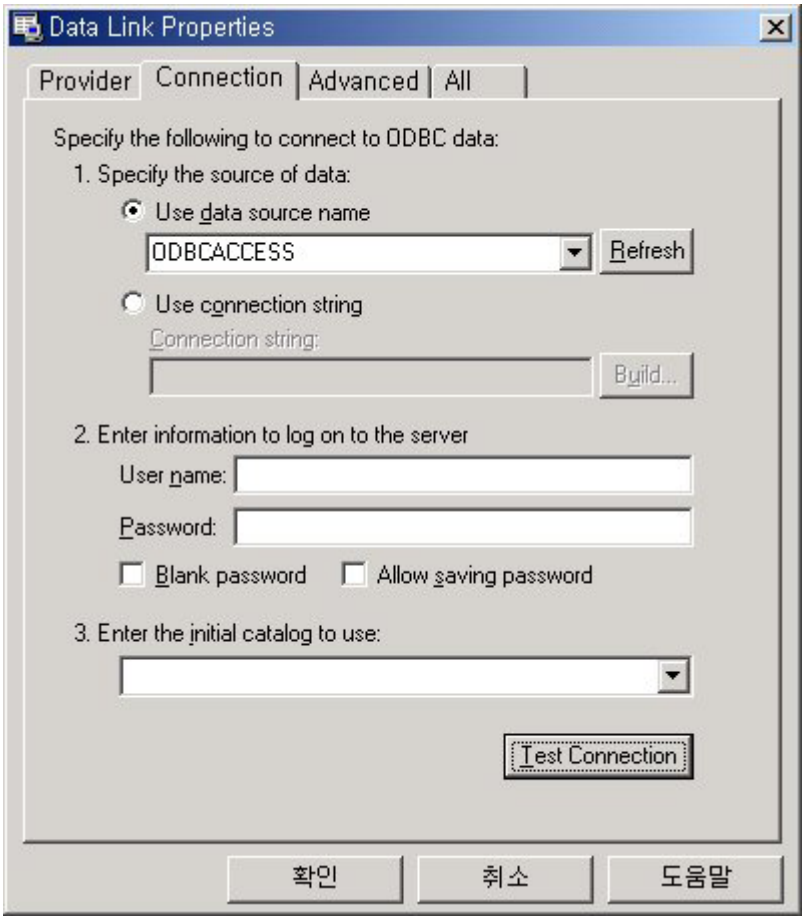
이전의 창으로 돌아와 확인 버튼을 누르면 아래 그림과 같이 ODBC용 Access Driver의 Data Source Name이 만들어 진다.



자 그럼 빌더의 TADOConnection Component로 돌아와 ConnectionString을 완성하자. SQL Server와 마찬가지로 ODBC용 OLEDB Provider를 선택한 후 Next 버튼을 누르자.



Data Source Name을 사용하므로 다음의 그림과 같이 만들어둔 Access용 ODBC Driver를 사용하면 된다.



Test Connection을 클릭하여 연결을 검사한다.

이와 같은 친숙하고 즐거운 창이 뜨면 연결에 성공한 것이며 우리의 레밍도 기쁠 것이다.

다음은 위와 같이 했을 때 얻어지는 ConnectionString이다.
Provider=MSDASQL.1;Persist Security Info=False;Data Source=ODBCACCESS

SQL Server 때와 같이 동일한 Provider에 단지 Data Source 항목의 값만 ODBC 용Access Driver Data Source Name으로 바뀐 것을 알 수 있을 것이다.

이것으로 지루하고 다소 복잡한 방법으로 ConnectionString을 얻는 과정을 SQL Server와 Access를 통해 알아보았다. 우리의 레밍은 연결은 성공 하였으나 지루하고 고된 여행에 지쳐 졸고 있다. 여기서 목적 중심의 사고를 약간 한다면 위의 다소 복잡한 설정 단계와 마우스를 여리고 작은 발로 부비는 고통과는 별도로 우리 레밍은 필요한 기반 설정에 맞는 ConnectionString만을 가지면 된다는 것이다. 이것은 동적으로 TADOConnection을 생성했을 때의 ConnectionString의 설정 방법과 같은 맥락인데 물론 Test Connection 버튼을 눌러 그 친근하고 즐거운 느낌이 나는 연결 성공 창을 볼 순 없지만 눈치가 빠르고 여러 번 고통을 느껴 봐서 통박을 깨우친 일부 레밍들은 연결 정보를 사용, 자기가 직접 ConnectionString을 만들어 직접 코드에 들이미는 놀라운 대담성을 보인다. 사실 연결을 동적으로 열거나 비주얼 하지 않은 어플리케이션 내에서는 이런 방법이 큰 도움이 된다. 자 그럼 그것에 대해 알아보자.

뭐 별게 있는 게 아니라 이제까지 얻은 ConnectionString을 일일이 코딩으로 쳐 주면 된다.
다음 코드는 이전 강의 연결의 생성과 해제부분의 소스인데 이번 강의의 내용을 끼워 넣었다 함 보자.

```
TADOConnection *objCon;  // Connection 객체 선언.

try
{
   objCon = new TADOConnection(NULL);
   // Connection 객체 인스턴스 생성.
   //.........................
   // 여러가지 세부설정을 한다.

   objCon->ConnectionString = 연결 스트링;

   objCon->Open();  // Connection 연결을 연다.
   //.........................
   // 여러가지 작업을 한다.

   if (objCon->State == (TObjectStates() << stOpen))
       objCon->Close(); // 연결을 닫는다.
}
__finally
{
   delete objCon;   // 혹은 objCon->Free();
   // 사용된 Connection 객체를 해제한다.
}
```

대략 간단한 구조는 위의 코드와 같다. 분명 이와 같은 방법으로 ConnectionString을 열면 일일이 누르고 설정할 필요가 없어진다. 종종 게으르고 무대포인 레밍중의 일부는 Native OLEDB Provider뿐 아니라 ODBC용 OLEDB Provider도 이런 방법으로 ConnectionString을 생성하는데 이러면 일일이 ODBC용 Data Source Name을 생성할 필요가 없어진다. 간단히 말해 DSN 없이 ODBC 접속을 사용하는 것이 된다. 이 방법의 이점은 ODBC를 통해 접속할 때 별다른 Setting이 필요 없고 모두 코딩상으로 설정이 해결된다는 장점이 있다. 자 그럼 ConnectionString을 알아보자. 연결 정보는 앞의 예제와 같다.

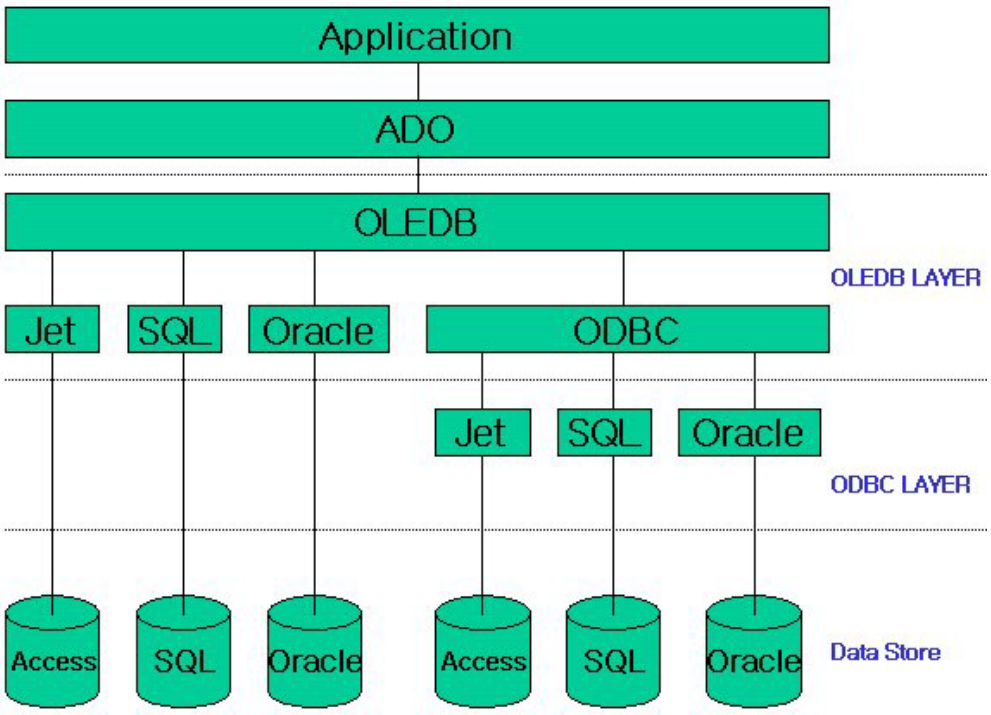
먼저 Access의 경우
Driver={Microsoft Access Driver (*.mdb)}; DBQ=D:₩CppBuilder₩Lecture₩2부₩db1.mdb

다음 SQL Server의 경우

Driver={SQL Server}; Server=SIGULS; Database=Pubs; UID=lemming; PWD=0000

여기서 주의 깊게 볼 부분은 두 가지 경우가 다 Provider가 아니라 Driver 항목이 사용되었다는 점이다. SQL Server의 경우 Driver={SQL Server} 대신 Provider=SQLOLEDB.1 을 사용할 경우 ODBC가 아니라 OLEDB를 사용해 연결하는 ConnectionString 임을 알 수 있다. ConnectionString을 생성할 때 OLEDB Provider들의 리스트를 보면 ODBC에 대한 OLEDB Provider가 항상 디폴트로 설정되어 있음을 보았을 것이다. 그러므로 여러분이 만약 Provider= 부분을 쓰지 않았거나 그대로 남겨 두었다면 자동적으로 여러분은 ODBC를 사용하게 되는 것이다. 앞에서 이전 강의의 서툰 그림을 들먹거리며 ADO Data Layer를 언급한 적이 있었는데 이제 그것을 정리하는 입장에서 개요가 아닌 이번 강좌의 내용에 맞는 그림으로 살펴 보도록 하자. 또 Provider와 Driver에 대해 자세하게 알아보자.

처음 강의에서도 이번 강의에서도 언급하지만 한 가지 중요한 사실은 OLEDB Provider가 ODBC를 포함한다는 것이다. 이 사실 때문에 OLEDB가 기존의 ODBC 데이터 소스들을 액세스할 수 있었고 우리는 앞의 예제에서 알아보았다. 그리고 이것의 가장 큰 이점이 ODBC가 OLEDB보다 더 많이 퍼져있기 때문에 OLEDB Provider들 보다 훨씬 더 많은 개수의 ODBC 드라이버들을 사용할 수 있다는 점이라는 것을 여러분도 볼 수 있었다. 이런 기능은 각 벤더들에서 OLEDB Provider들을 제공할 때 까지 우리가 별도로 기다리지 않더라도 데이터 원본에 대한 액세스를 할 수 있다는 의미이다. 또 Provider와 Driver를 혼동하지 않는 것이 중요하다. 자 그림을 보자.

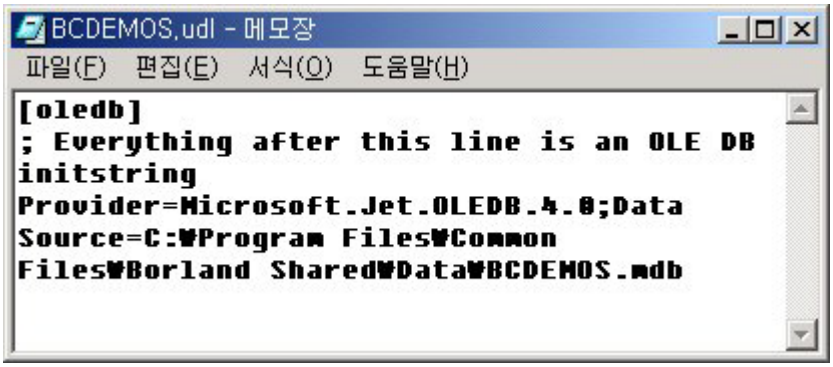


위의 그림은 앞에서 말한 그런 분할을 명확하게 나타내고 있는데 Provider들은 OLEDB 층에 있으며 드라이버들은 ODBC층에 있다. 이전의 예제들에서 살펴 본 것 같이 여러분이 ODBC 데이터 소스를 사용하고자 한다면 여러분은 ODBC에 대한 OLEDB Provider들을 사용하게 되고 그러면 ODBC는 적절한 ODBC Driver들을 사용하게 된다. 반대로 ODBC 데이터 소스를 사용할 필요가 없다면 적절한 native OLEDB Provider들을 사용할 수 있다.

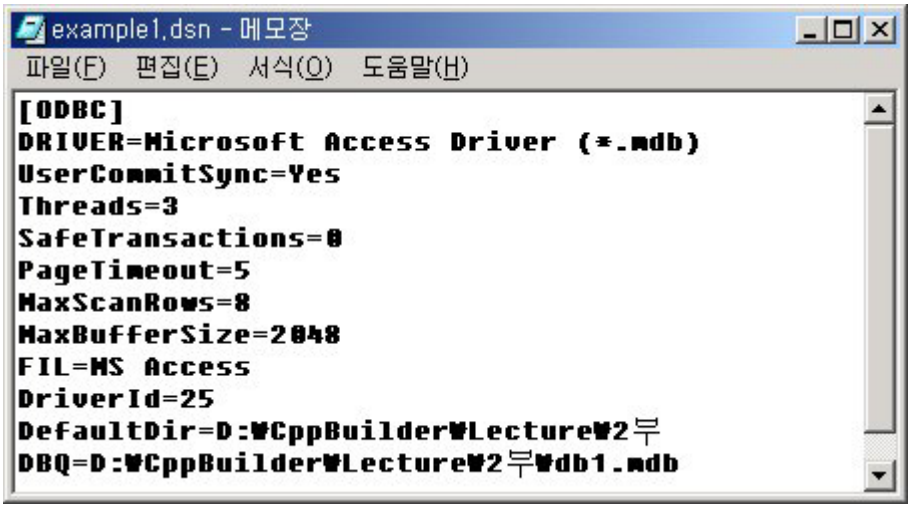
또한 첫번째 강의에서도 언급했지만 이 그림에서도 역시 볼 수 있는데 ODBC에 대해 OLEDB Provider들을 사용한다는 것은 또 하나의 층을 거쳐야 된다는 것을 분명히 볼 수 있다. 이런 이유로 동일한 데이터 원본에

액세스할 경우 ODBC에 대한 OLEDB Provider는 native OLEDB Provider들 보다 약간 느리게 작동한다.

그럼 마지막으로 갈림길로 돌아가서 ConnectionString에 Data Link File을 사용하는 방법에 대해서 알아보자. 이것은 특별한 것이 아니라 ConnectionString을 확장자가 udl인 텍스트 파일에 입력한 뒤 ConnectionString을 만드는 항목 중에서 Provider와 배타적인 File Name 항목을 이용, 데이터 원본과 연결을 얻는 것이다. 이런 데이터 링크 파일을 사용하게 되면 ConnectionString을 설정한 뒤에 나중에 ConnectionString이 변경되어도 어플리케이션을 재 Compile할 필요 없이 udl 파일 내의 ConnectionString만을 변경하면 된다. 이것은 분명 ADO 어플리케이션의 이식과 배포, 유지보수에 상당한 장점으로 작용한다. 반대로 단점도 존재하는데 바로 연결 정보 중에 데이터베이스 로그인 정보와 같은 보안에 중요한 정보가 드러나는 것에 관계된 문제이다. 이와 같은 득실의 문제는 프로젝트 중에 고려해야 될 중요한 문제이다. 필자의 경우 보안이 중요하지 않고 반대로 이식과 배포의 비중이 높은 간단한 로컬 데이터베이스를 사용하는 어플리케이션을 제작할 때 이 방법을 사용한다. 그리고 Web 어플리케이션의 경우 이 방법을 사용하려면 udl 파일을 Web Site 내의 URL로 mapping된 디렉터리에 두지 않는 것이 좋다. 자 그럼 구체적으로 살펴보자.

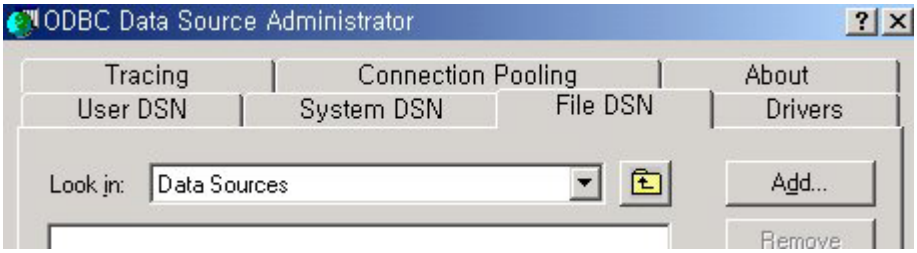


이전 예제에서 얻은 ConnectionString으로 udl 파일을 만드는 것은 여러분에게 맡긴다.



위의 BCDEMOS.udl은 C++ Builder가 설치되면 함께 설치되는 Default udl 파일이다. 파일의 내용에 ConnectionString을 볼 수 있을 것이다. 위의 2번째 그림은 필자가 ODBC Data Source Administrator의 File DSN에서 만든 example1.dsn의 내용이다.

특히 ODBC용 OLEDB Provider를 사용할 경우 File DSN을 만든 후 메모장으로 읽고 그 내용을 Copy하여 udl 파일을 만들면 쉬울 것이다. 아래의 그림을 보자.



Add 버튼을 눌러 ODBC Driver를 설정하고 저장하고자 하는 디렉터리 패스를 선택한 후 적당한 이름을 입력하면 dsn 파일이 만들어 진다. 그 내용을 Copy하여 새로운 udl 파일을 만든다. udl 파일을 만든 후 C++ Builder로 돌아가자.

ConnectionString을 생성하는 창에서 첫번째 항목인 Use Data Link File을 택하고 Browse 버튼을 눌러 위에서 생성한 udl 파일을 선택하면 된다.
OK 버튼을 누르면 얻어지는 ConnectionString은 다음과 같다. FILE NAME=C:₩Program Files₩Common Files₩System₩Ole DB₩Data Links₩BCDEMOS.udl
연결에 필요한 정보들은 모두 udl 파일 내에 있으므로 다른 항목에 대한 설정은 필요 없다. udl 파일은 기본적으로 텍스트 파일이므로 어플리케이션 내에서 TStringList 객체를 사용하여 동적으로 생성하는 등 다양한 응용 또한 가능하다. 그것은 전적으로 여러분의 몫일 것이다.

자 그럼 마칠 시간이다. 이번 강의는 생성된 연결 객체를 통해 데이터 원본에 접근하는 데 가장 핵심적인 설정인 ConnectionString에 대해 알아보았다. 이번 강의는 매우 중요하며 이후의 강의의 전제가 된다. 연결에 필요한 나머지 세부 설정들과 그에 관련한 TADOConnection 프로퍼티를 알아보고 직접 연결을 열어보자. 갈 길이 앞으로 멀다. 우리의 레밍을 격려해서 다음엔 조심스레 문을 열어보자.

Mortalpain