클라이언트/서버 데이터베이스 어플리케이션의 제작 (II)

이번 장에서는 앞 장에 이어서 클라이언트/서버 데이터베이스 어플리케이션을 제작할 때 고려해야할 여러가지에 대해서 설명할 것이다. 그 중에서도 공통적으로 접속을 관리하는 델파이의 컴포넌트 들의 사용법을 소개하고, 실제로 많이 쓰이고 있는 몇몇 DBMS 들의 특징과 이들을 다룰 때 주의해야 할 점에 대해서 알아볼 것이다. 지면에 여유가 있다면, 각각의 DBMS 에서 제공하는 SQL 문장의 독특한 점에 대해서도 다루면 좋겠지만 이를 언급하는데 따로 1 권의 책이 필요할 정도로 많은 분량이기 때문에, 여기서는 주로 서버에 접속하는 부분에 중점을 두고 설명할 것이다.

서버 로그인의 조절

대부분의 원격 데이터베이스 서버는 데이터베이스에 접근할 때 사용자 이름과 패스워드를 요구한다. 런타임에서 로그인에 대한 서버의 요구를 다루는 방법은 3 가지가 있다.
데이터베이스 컴포넌트의 LoginPrompt 프로퍼티를 True(디폴트)로 설정하면, 어플리케이션은 서버가 사용자 이름과 패스워드를 요구할 때 표준 로그인 대화상자를 보여주게 된다. 또 한가지 방법은 LoginPrompt 를 False 로 설정하고, 데이터베이스 컴포넌트의 Params 프로퍼티에서 사용자 이름과 패스워드를 설정하는 방법이다. 예를 들어,

```
USER NAME=SYSDBA
PASSWORD=masterkey
```

> 주의:
> Params 프로퍼티는 쉽게 볼 수 있기 때문에, 서버의 보안성이라는 측면에서 추천할 만한 방법이 아니다.

마지막으로 데이터베이스 컴포넌트의 OnLogin 이벤트에서 데이터베이스 Params 프로퍼티에 있던 내용을 이용하는 방법이다. OnLogin 의 LoginParams 의 Values 프로퍼티를 이용해서 다음과 같이 설정한다.

```
LoginParams.Values['USER NAME'] := UserName;
LoginParams.Values['PASSWORD'] := PasswordSearch(UserName);
```

런타임에서 데이터베이스 컴포넌트 만들기

런타임에서 데이터베이스 컴포넌트를 생성하려면 이름을 유일한 값으로 설정해야 하며, 세션과의 연관성을 가지게 해야 한다. 다음의 함수는 TempDatabase 라는 임시 데이터베이스를 이용해서 세션과 데이터베이스가 기존에 존재하지 않을 경우 새로운 세션과 데이터베이스 컴포넌트를 생성해 주는 예제이다.

```
function RunTimeDbCreate(const DatabaseName, SessionName: string): TDatabase;
var
  TempDatabase: TDatabase;
begin
  TempDatabase := nil;
  try
    {세션이 있으면 활성화, 없으면 새로운 세션을 생성}
    Sessions.OpenSession(SessionName);
    with Sessions do
      with FindSession(SessionName) do
        Result := FindDatabase(DatabaseName);
        if Result = nil then
          begin
           {새로운 데이터베이스 컴포넌트 생성}
           TempDatabase := TDatabase.Create(Self);
           TempDatabase.DatabaseName := 데이터베이스 Name;
           TempDatabase.SessionName := SessionName;
           TempDatabase.KeepConnection := True;
        end;
        Result := OpenDatabase(DatabaseName);
      end;
    end;
  except
    TempDatabase.Free;
    raise;
  end;
end;
```

다음의 코드는 이 함수를 이용해서 런타임에서 디폴트 세션의 데이터베이스 컴포넌트를 생성하는 역할을 한다. 이 때 데이터베이스의 이름이 중복되지 않도록 MyDbCount 라는 정수형 변수를 이용한다.

```
var
  MyDatabase: array [1..10] of  TDatabase;
  MyDbCount: Integer;
begin
  {MyDbCount 의 초기화}
  MyDbCount := 1;
  ...
  {런타임에서 데이터베이스 컴포넌트 생성}
  begin
    MyDatabase[MyDbCount]:=RunTimeDbCreate('MyDb'+IntToStr(MyDbCount),'');
    Inc(MyDbCount);
  end;
  ...
end;
```

원격 서버에 접속할 때 고려할 점

디자인 시에 커넥션 파라미터를 생성하거나 편집하는 방법에는 다음 3 가지 방법이 있다.

1. 데이터베이스 탐색기(Databse Explorer)나 BDE 관리 유틸리티 (BDE Admininstration utility)를 이용하는 방법
2. 오브젝터 인스펙터의 Params 프로퍼티를 더블 클릭하여 문자열 리스트 에디터(String List editor)를 호출하여 이용하는 방법
3. 데이터베이스 컴포넌트를 더블 클릭하여 데이터베이스 프로퍼티 에디터(Database Properties editor)를 호출하여 이용하는 방법

이들 방법은 모두 데이터베이스 컴포넌트의 Params 프로퍼티를 편집하게 된다. 여기에는 패스 정보, 서버 이름, 스키마의 캐쉬 크기, 언어 드라이버, SQL 쿼리 모드 등에 대한 정보가 들어 있다.
런타임에서 앨리어스 파라미터를 설정하려면 Params 프로퍼티를 직접 편집하면 된다.

세션의 관리

세션의 현재 상태를 결정하려면, Active 프로퍼티를 검사한다. Active 가 False 면 (디폴트) 세션과 연관된 모든 데이터베이스와 데이터 세트가 닫혀 있는 것이며, True 이면 열려 있는

것이다.
Active 를 True 로 설정하면 세션의 OnStartup 이벤트를 호출하며, NetFileDir, PrivateDir, ConfigMode 프로퍼티의 값을 설정한다. NetFileDir 과 PrivateDir 프로퍼티는 파라독스 테이블에 연결하기 위해 쓰이며, ConfigMode 는 BDE 가 세션에서 생성된 BDE 앨리어스를 어떻게 다룰 것인지를 결정한다.
일단 세션을 활성화한 뒤에는 데이터베이스 연결은 OpenDatabase 메소드를 통해 실시할 수 있다. 데이터베이스나 데이터 세트가 열려 있는 경우 세션 컴포넌트의 Active 프로퍼티를 False 로 설정해야 한다.

- 패스워드가 걸려있는 파라독스 테이블의 이용

세션 컴포넌트는 파라독스 테이블의 패스워드를 다루기 위해 4 개의 메소드와 1 개의 이벤트를 제공한다. 제공하는 메소드는 AddPassword, GetPassword, RemoveAllPasswords, RemovePassword 이며, 이벤트는 OnPassword 이다.
AddPassword 프로시져는 문자열을 파라미터로 넘기면 세션이 파라독스 테이블을 열 때 패스워드를 제공하여 바로 접근이 가능하게 된다. 만약에 패스워드가 걸려있는 파라독스 테이블에 접근할 때 AddPassword 를 호출하지 않고, OnPassword 이벤트 핸들러도 작성하지 않은 경우에는 패스워드를 입력하라는 대화 상자가 뜬다.
RemovePassword 는 삭제할 패스워드 문자열을 파라미터로 넘기면 이전에 메모리에 추가된 패스워드를 제거한다. RemoveAllPasswords 는 이전에 추가된 패스워드를 모두 제거한다. GetPassword 는 OnPassword 이벤트를 일으킨다. OnPassword 이벤트는 어플리케이션이 파라독스 테이블을 처음 열려고 할 때 일어난다. 이 이벤트에서 BDE 에 패스워드를 제공하고, 다루는 코드를 써준다. 이 때 델파이의 디폴트 작업은 대화상자를 띄워서 패스워드를 입력받는 것이다.

- 컨트롤 파일과 임시 파일 위치의 지정

NetFileDir 은 파라독스의 네트워크 컨트롤 파일인 PDOXUSRS.NET 의 위치를 지정한다. 이 파일은 네트워크 드라이브에서 파라독스 테이블을 공유하게 해주는 역할을 한다. 파라독스 테이블을 공유해야 하는 모든 어플리케이션은 반드시 같은 네트워크 컨트롤 파일의 디렉토리를 지정해야 한다 (보통은 네트워크 파일 서버의 디렉토리를 지정한다).
PrivateDir 은 BDE 에서 로컬 SQL 문장 등을 다룰 때 생성하는 임시 테이블을 저장할 디렉토리를 지정한다.

1. 컨트롤 파일 위치의 지정

델파이는 주어진 데이터베이스 앨리어스의 BDE 환경설정 파일에서 NetFileDir 의 값을 알아낸다. NetFileDir 값을 직접 설정하면, 이 값이 BDE 환경설정으로 기록된다.
다음 코드는 디폴트 세션의 NetFileDir 을 어플리케이션이 실행되는 디렉토리로 지정한다.

```
Session.NetFileDir := ExtractFilePath(ParamStr(0));
```

주의: NetFileDir 은 어플리케이션이 열린 파라독스 파일이 없을 때만 바꿀 수 있다.

2. 임시 파일 위치의 저장

PrivateDir 프로퍼티에 값이 지정되어 있지 않으면, BDE 는 현재의 디렉토리를 사용한다. 만약 어플리케이션이 네트워크 파일 서버에서 직접 실행될 경우 데이터베이스를 열기 전에 PrivateDir 을 사용자의 하드 디스크에 있는 디렉토리로 지정함으로써 어플리케이션의 퍼포먼스를 향상시킬 수 있다.
다음의 코드는 디폴트 세션의 PrivateDir 프로퍼티를 사용자의 C:₩TEMP 디렉토리로 지정한다.

```
Session.PrivateDir := 'C:₩TEMP';
```

- BDE 앨리어스 관리

1. 앨리어스의 visibility 지정

세션의 ConfigMode 프로퍼티는 어떤 BDE 앨리어스가 세션에서 사용가능한지를 결정한다. ConfigMode 는 어떤 형태의 세션들을 쓸 수 있는지를 결정하는 세트(set) 이다. 디폴트 값은 cmAll 로 [cfmVirtual, cfmPersistent]와 같은 의미이다. 이 때에는 세션에서 생성되거나, BDE 환경설정 파일에 있는 모든 앨리어스를 사용할 수 있다.
ConfigMode 프로퍼티의 주된 목적은 어플리케이션이 세션 레벨에서 앨리어스의 보이는 정도를 조절하기 위한 것이다. 예를 들어, ConfigMode 를 [cfmSession]으로 설정하면 세션에서 생성된 앨리어스만 쓸 수 있다.

2. 세션 앨리어스를 다른 어플리케이션이나 세션에서 사용하게 하려면 ...

세션에서 생생된 앨리어스는 BDE 에 의해 메모리에 복사되어 저장된다. 기본적으로 이 복사본은 생성된 세션에서만 쓰인다. 이 앨리어스를 같은 어플리케이션의 다른 세션에서 사용하려면 ConfigMode 프로퍼티가 cmAll 이거나 [cfmPersistent]여야 한다.

앨리어스를 모든 세션이나 어플리케이션에서 사용할 수 있게 하려면 세션의 SaveConfigFile 메소드를 사용해야 한다. SaveConfigFile 은 메모리의 앨리어스를 BDE 환경설정 파일에 기록하며, 이 파일을 사용하는 어플리케이션에서 읽을 수 있게 된다.

3. 앨리어스, 드라이버, 파라미터의 지정

세션 컴포넌트의 5 개의 메소드가 어플리케이션에서 BDE 앨리어스의 정보를 얻는데 쓰인다. 이들은 다음과 같다.

- GetAliasNames: 세션이 접근하는 앨리어스를 나열한다.
- GetAliasParams: 지정된 앨리어스의 파라미터를 나열한다.
- GetAliasDriverName:
  앨리어스가 사용하는 BDE 드라이버의 이름이 담긴 문자열을 돌려준다.
- GetDriverNames: 세션에서 사용가능한 모든 BDE 드라이버의 이름을 나열한다.
- GetDriverParams: 지정된 드라이버에 대한 드라이버 파라미터들을 돌려준다.

4. 앨리어스의 생성, 수정, 삭제

세션은 앨리어스를 생성, 수정, 삭제할 수 있다. AddAlias 메소드는 SQL 데이터베이스 서버에 대한 BDE 앨리어스를 생성하는 메소드이다. AddStandardAlias 는 파라독스, 디베이스, 아스키 테이블에 대한 새로운 BDE 앨리어스를 생성한다.
AddAlias 는 3 개의 파라미터를 가진다. 앨리어스의 이름을 가진 문자열, 사용할 SQL 링크 드라이버를 지정하는 문자열, 앨리어스에 대한 파라미터를 담은 문자열 리스트가 그것이다. 예를 들어, 다음의 문장은 인터베이스 서버에 접근하는 새로운 앨리어스를 추가한다.

```
var
  AliasParams: TStringList;
begin
  AliasParams := TStringList.Create;
  try
    with AliasParams do
      Add('OPEN MODE=READ');
      Add('USER NAME=TOMSTOPPARD');
      Add('SERVER NAME=ANIMALS:/CATS/PEDIGREE.GDB');
    end;
    Session.AddAlias('CATS', 'INTRBASE', AliasParams);
```

```
    .
    .
    .
  finally
    AliasParams.Free;
  end;
end;
```

AddStandardAlias 는 3 개의 문자열 파라미터를 가진다. 앨리어스의 이름, 파라독스 또는 디베이스 테이블의 패스, 테이블을 열때 사용할 디폴트 드라이버가 그것이다. 예를 들어, 다음 문장은 파라독스 테이블에 접근할 새로운 앨리어스를 생성한다.

```
AddStandardDriver('MYDBDEMOS', 'C:₩TESTING₩DEMOS₩', 'Paradox');
```

일단 앨리어스를 생성하면, 파라미터는 ModifyAlias 를 호출하여 수정할 수 있다. ModifyAlias 는 2 개의 파라미터를 가지는데, 하나는 앨리어스의 이름이고 다른 하나는 바꿀 파라미터와 그 값을 나열한 문자열 리스트이다. 다음 예제는 CATS 앨리어스의 OPEN MODE 파라미터를 READ/WRITE 로 바꿔 준다.

```
var
  List: TStringList;
begin
  List := TStringList.Create;
  with List do
  begin
    Clear;
    Add('OPEN MODE=READ/WRITE');
  end;
  Session.ModifyAlias('CATS', List);
  List.Free;
```

세션에서 생서왼 앨리어스를 삭제할 때 쓰는 메소드는 DeleteAlias 이다. DeleteAlias 는 삭제할 앨리어스의 이름을 파라미터로 한다.

주의: DeleteAlias 는 BDE 환경설정 파일에 기록된 앨리어스는 삭제하지 못한다. 만약 환경설정 파일에 기

록된 앨리어스까지 삭제하려면 DeleteAlias 호출 후 SaveConfigFile 메소드를 호출해야 한다.

- 데이터베이스 커넥션의 생성, 열기, 닫기

세션에서 데이터베이스 커넥션을 열려면 OpenDatabase 메소드를 사용한다. 이 메소드는 열어야 하는 데이터베이스의 이름을 파라미터로 넘기는데, 이름은 BDE 앨리어스이거나 데이터베이스 컴포넌트의 이름이어야 한다. 파라독스와 디베이스의 경우에는 이름이 패스를 포함한 이름일 수도 있다. 예를 들어, 다음 문장은 DBDEMOS 앨리어스가 가리키는 데이터베이스와의 커넥션을 연다.

```
var
  DBDemosData: TDatabase;
begin
  Session.OpenDatabase('DBDEMOS');
  ...
```

OpenDatabase 가 호출될 때마다 데이터베이스의 참조계수(reference count)가 1 씩 증가한다. 참조계수가 0 보다 큰 동안 데이터베이스는 열려 있다.
CloseDatabase 메소드를 사용하면 각각의 데이터베이스 커넥션을 닫을 수 있으며, Close 메소드를 사용하면 한 번에 모든 커넥션을 닫을 수 있다. CloseDatabase 를 호출하면 데이터베이스의 참조계수가 1 씩 감소하며, 이 수치가 0 이 되면 데이터베이스가 닫히게 된다. CloseDatabase 는 닫을 데이터베이스의 이름을 파라미터로 받는데, 이 값은 BDE 앨리어스, 데이터베이스 컴포넌트 이름 등이 될 수 있다. 예를 들어, 다음 문장은 DBDEMOS 앨리어스에 지정된 데이터베이스 커넥션을 닫는다.

```
CloseDatabase('DBDEMOS');
```

지정된 데이터베이스 이름이 임시 데이터베이스 컴포넌트와 연관되어 있고, 세션의 KeepConnections 프로퍼티가 False 이면, 임시 데이터베이스 컴포넌트는 메모리에서 해제되며, 커넥션이 종료된다.
또한, 데이터베이스 컴포넌트가 미리 선언되어 있고, 인스턴스화 되어 있으며, 세션의 KeepConnections 프로퍼티가 False 이면, CloseDatabase 는 데이터베이스 컴포넌트의 Close 메소드를 호출하고 커넥션을 종료한다.
모든 데이터베이스 커넥션을 닫는 방법은 세션의 Active 프로퍼티를 False 로 설정하거나, 세션의 Close 메소드를 호출하면 된다.

세션의 추가

디자인 시에 세션을 추가하는 것은 쉽다. 단지 데이터 모듈이나 폼에 세션 컴포넌트를 위치시키고, 프로퍼티를 설정하고 이벤트 핸들러를 작성하면 된다. 런타임에서 세션을 생성하고 프로퍼티를 설정, 메소드를 호출하려면 다음과 같이 해야 한다.

1. TSession 변수를 선언한다.
2. Create constructor 를 호출하여 새로운 세션을 인스턴스화 한다. 이렇게 하면 세션에 대한 데이터베이스 컴포넌트들과 BDE 콜백(callback)의 빈 리스트가 작성되고, KeepConnections 프로퍼티가 True 로 설정된다. 어플리케이션의 세션 리스트에 세션이 추가된다.
3. SessionName 프로퍼티에 적절한 이름을 설정한다.
4. 세션을 활성화하고, 프로퍼티를 설정한다.

다음의 코드는 이런 단계를 잘 밟은 샘플 코드이다.

```
var
  SecondSession: TSession;
begin
  SecondSession := TSession.Create;
  with SecondSession do
    try
      SessionName := 'SecondSession';
      KeepConnections := False;
      Open;
  end;
  .
  .
  .
    finally
      SecondSession.Free;
  end;
end;
```

TSessionList 의 OpenSession 메소드를 이용해서 세션을 생성하고 열 수 있다. 이렇게 OpenSession 을 사용하는 것이 Create 를 호출하는 것보다 안전한데, 이는 OpenSession 이

이미 존재하지 않는 세션만을 생성하기 때문이다.

세션의 여러 데이터베이스 컴포넌트에 대한 접근

세션의 Databases, DatabaseCount 프로퍼티는 세션과 연관된 데이터베이스 컴포넌트에 접근할 수 있게 해준다.
Databases 프로퍼티는 세션과 연관된 활성화된 데이터베이스 컴포넌트들의 배열이다. DatabaseCount 프로퍼티와 함께 사용하여 모든 활성화된 데이터베이스 컴포넌트에 영향을 줄 수 있다. DatabaseCount 는 정수형 프로퍼티로 현재 세션과 연관된 활성화된 데이터베이스의 수를 나타낸다.
예를 들어, 다음의 코드는 각각의 활성화된 데이터베이스의 KeepConnection 프로퍼티를 True 로 설정한다.

```
var
  MaxDbCount: Integer;
begin
  with Session do
    if DatabaseCount > 0 then
      for MaxDbCount := 1 to DatabaseCount do
        Databases[MaxDbCount].KeepConnection := True;
  end;
end;
```

여러 세션의 관리

멀티 스레드를 사용하여 데이터베이스 조작을 하는 어플리케이션을 제작하려면, 각각의 스레드에 추가적인 세션을 생성해야 한다.

> 주의:
> 세션 컴포넌트를 추가할 때 반드시 SessionName 프로퍼티는 유일한 값으로 설정해서 디폴트 세션의 SessionName 프로퍼티와 혼동되지 않도록 해야 한다.

세션 컴포넌트를 디자인 시에 배치할 때는 어플리케이션이 동작할 때 몇 개의 스레드를 사용할 지 대강 짐작하고 있어야 한다. 그러므로, 세션을 동적으로 생성하는 것이 바람직한 경우가 많은데, 이렇게 하려면 런타임에서 Sessions.OpenSession 을 호출해야 한다.
Sessions.OpenSession 메소드는 세션의 이름을 파라미터로 받는다. 다음의 코드는 동적으

로 새로운 세션을 생성하고 활성화 시킨다.

```
Sessions.OpenSession('RunTimeSession' + IntToStr(Sessions.Count + 1));
```

이 문장에서 새로운 세션이 생성될 때마다 현재 세션의 수를 이름에 추가하여 세션의 이름이 유일한 값을 가지도록 하고 있다.
Sessions 는 TSessionList 의 컴포넌트로 데이터베이스 어플리케이션에 의해 자동으로 인스턴스화 된다. 다음 테이블에 TSessionList 컴포넌트의 프로퍼티와 메소드를 정리해 보았다.

프로퍼티, 메소드	목 적
Count	세션의 수를 돌려 준다 (active, inactive 포함).
FindSession	지정된 이름을 가진 세션을 찾아서 세션 컴포넌트의 포인터를 돌려준다. 찾는 세션이 없으면 nil 을 돌려주며, 파라미터에 아무것도 쓰지 않으면 디폴트 세션의 포인터를 돌려준다.
GetSessionNames	현재 인스턴스화된 세션 컴포넌트의 이름들을 문자열 리스트(string list)에 담아서 돌려준다. 최소한 'Default'는 돌아온다.
List	지정된 세션 이름을 가진 세션 컴포넌트를 돌려주며, 해당되는 것이 없으면 예외가 발생한다.
OpenSession	새로운 세션을 생성하고, 활성화 한다.
Sessions	세션 리스트에 Ordinal 값으로 접근할 수 있다.

세션의 정보 이용

세션에는 세션과 데이터베이스 컴포넌트의 정보를 알아낼 수 있는 메소드가 있다. 다음 테이블에 이들 메소드를 정리하였다.

메소드	목 적
GetAliasDriverName	데이터베이스의 지정된 앨리어스의 BDE 드라이버 정보 얻기
GetAliasNames	데이터베이스의 BDE 앨리어스의 리스트 얻기
GetAliasParams	데이터베이스의 지정된 BDE 앨리어스의 파라미터 목록 얻기
GetConfigParams	BDE 환경설정 파일에서 특정 환경설정 정보 얻기
GetDatabaseNames	현재 사용되는 TDatabase 컴포넌트의 이름과 BDE 앨리어스의 목록 얻기
GetDriverNames	현재 인스톨된 BDE 드라이버의 이름들 얻기
GetDriverParams	지정된 BDE 드라이버의 파라미터 리스트 얻기

GetStoredProcNames	지정된 데이터베이스의 모든 stored procedure 의 이름 얻기
GetTableNames	지정된 데이터베이스의 지정된 패턴과 일치하는 테이블들의 이름얻기

GetAliasDriverName 을 제외하고, 문자열 리스트의 세트를 리턴값으로 받는다 (GetAliasDriverName 은 단일 문자열이다.). 예를 들어, 다음 코드는 디폴트 세션의 모든 데이터베이스 컴포넌트와 앨리어스의 이름을 얻는다.

```
var
  List: TStringList;
begin
  List := TStringList.Create;
  try
    Session.GetDatabaseNames(List);
    .
    .
    .
  finally
    List.Free;
  end;
end;
```

데이터베이스 커넥션의 검색

세션의 FindDatabase 메소드를 사용해서 데이터베이스를 검색할 수 있다. 이 메소드는 찾고자 하는 데이터베이스의 이름을 파라미터로 받는다. 다음의 예제에서는 디폴트 세션에서 DBDEMOS 앨리어스를 사용하는 모든 데이터베이스 컴포넌트를 검색하며, 찾지 못하면 이를 생성하고 연다.

```
var
  DB: TDatabase;
begin
  DB := Session.FindDatabase('DBDEMOS');
  if DB = nil then                                    //데이터베이스가 세션에 없다.
    DB := Session.OpenDatabase('DBDEMOS');   //생성하고 open
  if Assigned(DB) and DB.Active then
  begin
```

```
    DB.StartTransaction;
    .
    .
    .
 end;
end;
```

인터베이스와 델파이

가장 먼저 알아볼 DBMS 는 Inprise 의 인터베이스이다. DBMS 자체로도 마켓 쉐어가 그다지 높다고 볼 수는 없지만, 싼 가격과 Inprise 에서 제작했다는 장점 때문에 델파이로 개발되는 프로젝트에서는 채택되는 경우가 많은 서버이다.
일반적인 DBMS 와 달리 인터페이스는 분리된 데이터베이스 접속 요소를 가지고 있지 않기 때문에, 인터페이스 서버에 접근하기 위해서는 이를 따로 정의해야 한다. 원격 인터베이스 서버에 접속하려 하면, 요구되는 접속 정보는 사용하는 프로토콜에 따라 다르다. TCP/IP 를 이용하는 경우 HOSTS 파일이 반드시 서버에 대한 레퍼런스를 포함해야 한다. 그 내용은 '123.33.44.12 marketing'과 같은 형태이면 된다. 또한, TCP SERVICES 파일에 'gds_db 3050/tcp'와 같이 인터베이스 접근 프로토콜을 지정해야 한다.
이런 작업은 인터베이스를 설치하면 자동적으로 이루어지며, 일단 TCP/IP 접근이 이루어지면 WISQL 등의 인터베이스 도구를 이용하여 접속이 가능하다.

- BDE 앨리어스의 설정

데이터베이스 서버에 접속이 가능해지면 BDE 앨리어스를 만들 수 있는데, 일단 BDE Administrator 유틸리티나 델파이 SQL 탐색기를 띄우고 앨리어스 설정 대화 상자를 띄운 뒤에 앨리어스 type 에서 INTRBASE 를 선택한다. 새로운 앨리어스가 생성되면 적당한 이름을 부여하고, Definition 탭에서 파라미터를 설정한다. SERVER NAME 파라미터를 서버의 이름과 접속하고자 하는 데이터베이스 파일 이름으로 설정하는데, 다음과 같은 형태를 가지게 된다.

```
Server:/data/인터베이스/sample.gdb
```

여기에서 Server 는 서버의 이름이며 나머지 부분은 데이터베이스 파일의 패스이다. 옵션으로 USER NAME 파라미터를 데이터베이스 사용자 이름으로 설정할 수 있다. 여기에서 설정한 사용자 이름은 델파이가 로그-인 대화상자의 디폴트 값으로 사용된다.

- 인터베이스 접속에 문제가 있을 때에는 …

델파이 어플리케이션을 이용해 인터베이스 서버에 접속할 때 문제가 있을 때에는 다음과 같은 방법으로 문제 해결을 시도하는 것이 좋다.
먼저, 인터베이스에서 제공되는 인터베이스 통신 진단 유틸리티(인터베이스 Communication Diagnostics utility)를 이용하여 데이터베이스 서버에 접속을 시도해 본다. DB 커넥션 페이지에 있는 Test 버튼을 클릭해서 접속이 가능한 경우에는 BDE 앨리어스의 설정이 잘못된 것이 접속이 안되는 원인일 가능성이 높다. 그러므로, BDE 관리자를 이용해서 앨리어스 설정이 잘 되었는지 알아보는 것이 좋다. 접속이 안되는 경우에는 NetBEUI 나 Winsock 페이지에서 Test 버튼을 클릭해서 접속을 시도한다. 이 경우에 접속이 된다면 지정된 내용에 문제가 있다는 의미이다.
다른 방법으로는 인터베이스의 WISQL 유틸리티를 이용하여 서버에 접속을 시도해 본다. 델파이 어플리케이션은 접속에 문제가 있는데, 여기에서는 접속이 된다면 BDE 앨리어스 설정의 문제로 보면 된다. BDE 앨리어스 문제를 해결할 때에는 보통 SERVER NAME 파라미터가 잘못된 경우가 많다.
WISQL 도 접속에 실패하는 경우에는 프로토콜 문제일 가능성이 높다. 그러므로, TCP/IP 를 사용한다면 ping 을 이용하여 호스트 컴퓨터에 접속을 시도한다. 그리고, NT 기반의 서버에서 명명된 파이프(named pipe)를 이용하는 경우라면 실행 메뉴를 이용하여 'net view ₩₩servername' 명령을 수행해 본다. 여기에서 servername 은 SQL 서버가 동작하는 NT 기계의 이름이다. 이 명령이 성공적으로 수행되면 'netuse ₩₩servername₩IPC$' 명령을 실행해 본다. 여기서 실패하는 경우라면 NT 시스템 설정에 문제가 있다는 의미이다. TCP/IP 를 사용하고, ping 도 정상적으로 동작하는 경우라면 아마도 HOSTS 파일의 내용이 문제가 있을 것이다. 이 파일의 내용이 제대로 설정되어 있는지 확인하기 바란다.
ping 이 안되는 경우라면 이것은 네트워크 문제로 간주하면 된다. ping 은 되는데, WISQL 로 접속이 안되고 HOSTS 파일에도 문제가 없는 경우에는 telnet 을 이용해서 접속을 시도해 본다. telnet 접속이 되지 않는다면 이것은 서버 기계의 inet 데몬이 제대로 동작하지 않는 것이다. telnet 접속이 되는데 WISQL 접속이 되지 않는 경우에는 인터베이스 서버 설치가 제대로 안되었을 가능성이 높다.

오라클과 델파이

오라클은 현재 전세계적으로 가장 많이 팔린 RDBMS 이다. 최근에 발표된 오라클 8 의 경우에는 ORDBMS 로서의 형태를 가지게 되어, 중첩된 테이블과 ODB 의 장점이 객체 연결 기능 등을 제공한다.
여기서는 오라클 서버에 접속하는 방법을 기준으로 간단하게 설명하고자 한다. 그렇지만, 오라클을 서버로 해서 개발을 하는 경우에는 오라클에 대한 내용을 보다 자세하게 이해하고

개발을 하면 보다 강력한 성능을 이용할 수 있다.
오라클에 접속하기 위해서는 오라클의 SQL Net 접속 소프트웨어를 설치하고 환경 설정을 해야 한다. 이를 설치하기 위해서는 단순히 오라클의 ORAINST 설치 프로그램을 실행하면 된다. 환경 설정을 하기 위해서는 먼저 데이터베이스 앨리어스를 정의해야 한다. 여기서 앨리어스는 데이터베이스 서버에 접속하기 위해 필요한 모든 정보를 포함한다. 이 앨리어스는 오라클 드라이버에 대해서만 동작하는 것으로, 델파이의 BDE 앨리어스와는 다르다.
오라클 데이터베이스 앨리어스를 SQL Net 환경설정 프로그램을 이용해서 환경 설정을 하려면, SQL Net Easy Configuration 프로그램을 시작하고 Add Database Alias 를 선택한 뒤 OK 버튼을 클릭한다.
다음 대화상자에서는 데이터베이스 앨리어스를 선택할 수 있는데, 여기에서 사용하고자 하는 데이터베이스 앨리어스를 선택한다. 만약 새로운 앨리어스를 추가하고자 할 때에는 새로운 이름을 적어 넣고 OK 버튼을 클릭한다.
그 다음에는 네트워크 프로토콜을 선택할 수 있다. 이렇게 네트워크 프로토콜을 선택한 뒤에는 서버에 대한 추가적인 정보를 주어야 한다. TCP/IP 를 선택한 경우라면 HOSTS 파일에 나타날 서버의 호스트 이름과 IP 주소를 입력해야 한다. SPX 를 선택한 경우에는 SPX 서비스 이름을 대신 사용한다.
마지막 대화상자에서는 데이터베이스 앨리어스에 대한 접속 정보가 표시된다. OK 를 클릭하면 데이터베이스 앨리어스가 추가된다. SQL Net configuration 프로그램의 메인 메뉴로 돌아가면, Exit 를 선택하고 어플리케이션을 닫는다.
데이터베이스 앨리어스가 환경설정된 후에는 이를 이용하여 데이터베이스 서버에 접속을 시도할 수 있다. 이를 위해서는 오라클의 SQL Plus 유틸리티를 이용하여 다음과 같이 접속을 시도할 수 있다.

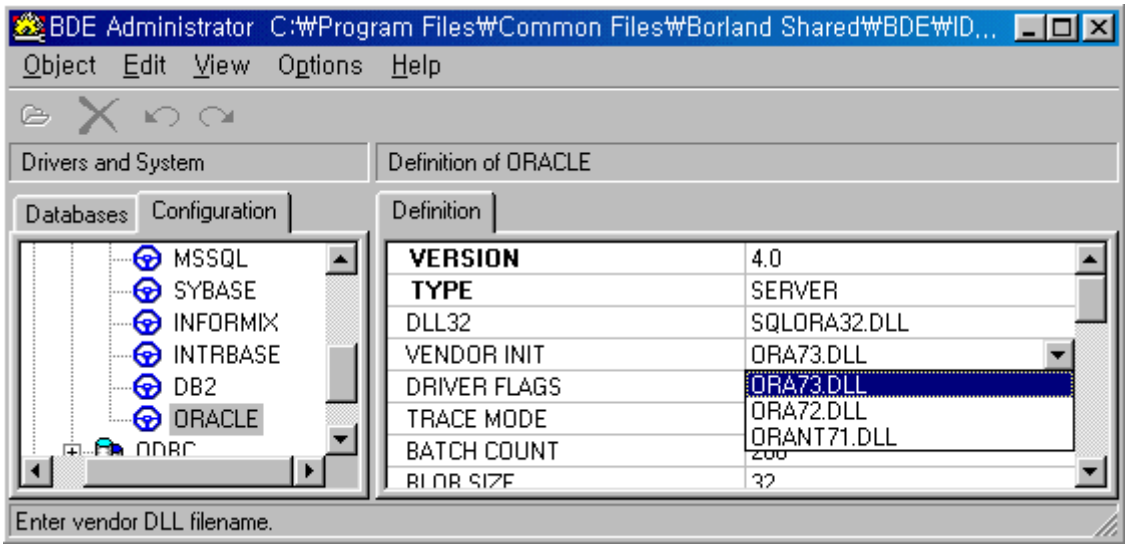
sqlplus USERNAME/PASSWORD@ALIASNAME @<file>.<ext>

USERNAME 과 PASSWORD 에는 사용하고자 하는 사용자 이름과 패스워드를 입력하고 ALIASNAME 에는 데이터베이스 앨리어스의 이름을 사용한다.
이 밖에도 오라클의 TNSPING 프로그램을 이용하여 서버 접속을 테스트할 수 있다. 이 프로그램은 ping 과 사용법이 비슷하다.

- BDE 앨리어스의 설정

서버에 접속이 가능하면, BDE 앨리어스를 만든다. BDE 관리자를 이용하면 되는데, 인터베이스에 비해 다소 복잡한 절차를 거친다.
앨리어스를 만들기 전에 먼저 Configuration 탭의 Drivers|Native|Oracle 을 선택하고, 오른쪽 탭의 VENDOR INIT 항목에서 다음과 같이 적절한 클라이언트를 설정한다. 예를 들

어 서버가 오라클 7.3 이라면 ORA73.DLL 을 선택해야 한다.



그 밖에도 이 곳에서 여러 항목을 변경할 수 있다. 일단 드라이버의 설정이 끝났으면, 데이터베이스 탭에서 앨리어스 type 이 ORACLE 인 새로운 앨리어스를 추가한다. 추가한 앨리어스에 대한 설정만 하면 BDE 앨리어스의 설정이 종료되는데 그 중에서 SERVER NAME 항목과 NET PROTOCOL 항목은 SQL Net configuration 프로그램에서 생성한 앨리어스와 네트워크 프로토콜로 설정한다. 또한, NAME 파라미터를 데이터베이스 서버에 접속하고자 하는 사용자의 이름으로 설정하면 델파이가 디폴트 로그-인 대화상자를 띄울 때 이 이름을 디폴트로 사용한다.

- 오라클 접속에 문제가 있을 때에는 ...

먼저 오라클의 SQL Plus 유틸리티를 이용하여 서버에 접속을 시도해 본다. 여기서 접속이 되지만 델파이 어플리케이션에서 접속이 되지 않는다면 이는 BDE 앨리어스 환경 설정의 문제로 봐야 한다. 만약 SQL Plus 유틸리티로 접속이 되지 않으면, 프로토콜의 문제를 의심해야 한다. 서버와의 접속에 TCP/IP 를 사용한다면 ping 을 이용하여 해당 서버와의 접속이 되는지 테스트하고, 이때 호스트 이름으로는 접근이 안되는데 IP 주소로는 접속이 된다면 HOSTS 파일에 문제가 있는 것이므로 이를 수정한다. SPX 프로토콜을 사용하는 경우에는 서버의 SPX 서비스 번호를 제대로 설정해야 한다.
TCP/IP 를 사용하는 경우에 ping 이 제대로 되지 않는다면 네트워크 상의 문제로 봐야 한다. SQP Plus 로 접속은 되지 않는데, ping 이 잘 된다면 오라클 홈 디렉토리가 사용하고자 하는 패스르 지정되었는지 알아본다. 패스가 정확하다면 (홈 디렉토리)₩NETWORK ₩ADMIN₩TNSNAMES.ORA 파일이 제대로 설정되었는지 알아보아야 한다. 이 파일은 텍스트 파일이기 때문에 쉽게 편집할 수 있지만, 직접 편집하기 보다는 SQL Net

configuration 프로그램을 이용하여 편집하는 것이 좋다.
설정에 문제가 없다고 판단되면, 프로토콜을 한 번 변경해보는 것이 좋다.

- 그 밖에 …

오라클은 DBMS 시장에서 마켓 쉐어 1 위의 제품인 만큼, 접속 차원에서의 문제를 제외하고도 유용한 여러가지 정보가 있다. 여기에 대해서 설명할 것이다.
보통 RDB 로 시스템을 구성하게 되면 RDB 상에 무수히 많은 데이터베이스가 존재할 수 있기 때문에, 이를 정리하여 주는 관리자의 필요성이 크게 대두된다. 그렇기 때문에, 데이터베이스를 만들 경우에 해당 데이터베이스의 관리자를 정하게 된다. 이때, 단독 관리자도 존재하지만 작업내용에 따라 일을 분담하여 작업하게 되는 복수의 관리자가 존재할 수도 있다. 오라클의 경우 SQL 이나 DBA 가 CREATE DATABASE 명령을 통하여 정상적으로 데이터베이스를 생성할 때, 다음의 SYS 와 SYSTEM 이라는 사용자가 자동적으로 등록된다.

1. SYS: 데이터베이스의 데이터 사전 소유자
2. SYSTEM : SQL Forms 등의 툴을 위한 데이터 사전 소유자

이제 SYS 와 SYSTEM 은 새로운 DBA(관리자)권한을 갖는 사용자를 만들 수 있는 권리를 가지게 된다. 그런데, 왜 이런 식으로 두 개의 관리자가 존재하는 것일까? 그것은 DB 시스템의 관리와 어플리케이션 관리자를 다르게 보기 때문이다. 보통 관리자라 함은 SYSTEM 관리자로써 DBA 작업을 수행하게 된다.
SYS 는 CHANGE_ON_INSTALL, SYSTEM 은 MANAGER 라는 디폴트 패스워드를 가지고 있다. 그렇기 때문에, 보안을 위해서 이를 빨리 교체해 주는 것이 좋다.

> 참고:
> 인터베이스의 경우 SYSDBA 와 MASTERKEY 로 구분되는 기본적인 DBA 의 설정을 갖는다. 그리고 MS-SQL 서버의 경우 SA 로 설정되는 기본적인 DBA 관리자가 존재한다.

SYSTEM 의 패스워드를 변경하기 위한 명령어는 'ALTER USER SYSTEM IDENTIFIED BY 새로운 패스워드' 이다.
참고로 오라클을 설치할 경우에 언어 설정에서 한국어로 한정하면 보통 클라이언트 제품에서 한글을 지원하지 않고, 영어만 지원하는 경우에 충돌의 우려가 있으므로 이 경우에는 영어로 설정하여 사용하는 것이 좋다.

마이크로소프트 SQL 서버와 델파이

마이크로소프트 SQL 서버는 저렴한 가격과 윈도우 NT 서버의 보급으로, 최근 들어 가장 가파른 마켓 쉐어의 상승을 가져오고 있는 제품이다. 앞으로도 오라클과 함께 시장을 양분할 것으로 예상되는 대표적인 DBMS 이다.

- BDE 앨리어스의 설정

BDE 앨리어스를 새로 만들 때에는 앨리어스 type 으로 MSSQL 을 선택하는 것을 제외하고는 다른 경우와 같다. 생성된 앨리어스의 SERVER NAME 항목에 SQL 서버의 이름을 설정해야 하는데, 이 이름은 SQL Client Configuration 유틸리티에서 사용된 이름과 일치해야 한다. USER NAME 항목을 설정하면 디폴트 로그-인 대화 상자에 설정된 이름을 사용할 수 있다.
그 밖에 SQL 서버에서 사용한 몇 가지 항목들이 있다.

1. APPLICATION NAME

이 항목은 SQL 서버의 sysprocesses 테이블에 사용될 프로세스의 이름을 나타나게 할 수 있다. 이렇게 하면, 어플리케이션 프로세스가 서버에서 다른 프로세스와 구별이 가능하다.

2. BLOB EDIT LOGGING

이 값을 False 로 설정하면 BLOB 필드에 변화가 생긴 내용을 logging 하지 않도록 할 수 있다. 이렇게 함으로써 BLOB 저장 요구를 최소화하고, 수행 성능을 향상시킬 수 있다. 이 방법은, BLOB 데이터를 SQL 서버의 벌크 복사 장치(bulk copy facility)를 이용하여 전송하므로 목적 데이터베이스에 select into/bulk copy 문장을 설정해야 한다. select into/bulk copy 는 sp_dboption 저장 프로시저를 이용하여 사용할 수 있다.

3. CONNECT TIMEOUT, HOST NAME

CONNECT TIMEOUT 항목은 클라이언트가 접속을 위해 대기하는 시간을 한정할 수 있다. 디폴트는 60 초로 설정되어 있는데, 경우에 따라서 이 값을 변경할 수 있다. 그리고, HOST NAME 항목을 현재의 컴퓨터로 설정하면 SQL 서버의 sysprocesses 테이블에 실행되는 호스트의 이름을 나타나게 할 수 있다. 이렇게 하면, sp_who 저장 프로시저를 이용해서 접속되어 있는 컴퓨터 들을 확인할 수 있다.

4. DATABASE NAME

이 항목은 접속할 SQL 데이터베이스의 이름을 설정할 때 사용한다. 이 값을 비워두면 서버의 디폴트 데이터베이스가 선택된다. 그렇지만, 이 값을 선택하여 데이터베이스를 명확히 하는 것이 좋다.

5. TDS PACKET SIZE

이를 이용하여 Tabular Data Stream(TDS) 패킷의 크기를 지정할 수 있다. TDS 는 SQL 서버가 클라이언트와 데이터를 교환할 때 사용하는 패킷 프로토콜이다. 이 값은 0~65535 까지 가능하지만, 실제로는 512~32767 까지의 값이 사용된다. 디폴트 값은 4096 이며, 보통 4096~8192 사이의 값의 수행 성능이 가장 높다.
SQL 서버의 sp_configure 저장 프로시저를 이용하면 서버에서 지원하는 현재의 최대 패킷 크기를 결정할 수 있다.

- 클라이언트 접속의 설정

SQL 서버의 클라이언트 접속은 SQL Client Configuration 유틸리티를 이용하여 환경 설정을 할 수 있다. 이 유틸리티를 클라이언트에서 실행하고 Advanced 탭을 선택한 뒤, 서버 엔트리 박스에서 서버의 이름을 설정한다. 그리고, Net Library Configuration 에서 사용할 디폴트 네투워크를 선택해야 하는데, 보통 명명된 파이프(named pipe)나 TCP/IP 중 하나를 선택하게 된다.
명명된 파이프를 사용하는 경우에 서버의 이름으로 컴퓨터 이름을 설정하는 것이 좋다. 그리고, TCP/IP 를 사용할 경우에는 IP 주소나 포트 번호 등의 정보를 따로 설정해야 한다.

- SQL 서버 접속에 문제가 있을 경우에는 ...

먼저 마이크로소프트의 ISQL/w 유틸리티를 이용하여 서버에 접속을 시도한다. 여기서 접속이 되지만, 델파이 어플리케이션에서 접속이 되지 않는다면 지금까지와 같이 BDE 앨리어스 설정이 잘못된 경우이다. ISQL/w 을 이용해도 접속이 되지 않으면, ping 을 이용하여 접속을 시도한다. 다른 내용은 모두 오라클의 경우와 같으므로 이를 참고하기 바란다.

정 리 (Summary)

이번 장에서는 클라이언트/서버 데이터베이스 어플리케이션을 개발할 때 알아야 할 몇 가지 컴포넌트의 사용 방법과 접속 관리, 주요 DBMS 에 접속할 때 알아야할 점 등에 대해서 알아보았다. 다음 장에서는 데이터베이스 어플리케이션을 개발할 때 도움이 되는 여러가지 간단한 팁들에 대해서 알아볼 것이다.