AnyDac gebruiken in Delphi

Michaël Van Canneyt

20 maart 2012

Samenvatting

Anydac is een verzameling data componenten voor Delphi. De componenten staan toe verbinding te maken met een databank zonder dat daar externe DLLs voor nodig zijn. Alle benodigde code wordt in de applicatie gelinkt. Buiten een TDataset gebaseerd model om data op te halen en te bewerken, zijn er flink wat extra componenten beschikbaar.

1 Introduction

De meeste - indien niet alle - programmas beheren een databank. Delphi programmeUrs zijn verwend met een groot gamma aan data componenten. De Delphi installatie zelf bevat 2 databank technologieen: De BDE (niet meer ondersteund) en de aanbevolen technologie: dbExpress. Met dbEXpress kan elke SQL databank gebruikt worden. Bovendien bevat Delphi ADO componenten, om de ADO technologie van Microsoft te gebruiken. Elke databank met een ODBC driver kan daardoor gebruikt worden.

Als geen van deze technologieen voldoening geeft, kan ZeosLib gebruikt worden, een reeks van open source componenten die ook een groot deel RDBMS databanken kan bedienen.

In dit artikel wordt een alternatief voor al deze mogelijkheden voorgesteld: AnyDAC. Anydac is een reeks componenten om verbinding te maken met een databank. Het is een commercieel product, gemaakt en verdeeld door DA-Soft. Het is beschikbaar op hun website:

`http://www.da-soft.com/`

AnyDAC is een uitgebreide verzameling componenten die de BDE, dbExpress, Zeos en ADO kan vervangen: Het verschaft directe toegang tot een groot aantal databanken: niet alleen open source databanken zoals Firebird, Interbase, SQLIte, MySQL, PostgreSQL en Berkely DB, maar ook commerciele databanken zoals Oracle, DB2, MS-SQL server, MS-Access, Sybase en Blackfish SQL server. Er zijn ook componenten voor flat-file databanken en in-memory data. Bovendien is er een component beschikbaar die toestaat data om te zetten van de ene naar de andere databank.

Naast toegang tot de data in de RDBMS, zijn er ook wat hulp componenten beschikbaar voor het aanmaken en terugzetten van backups - voor de databanken die dit ondersteunen. Voor databanken die een event (gebeurtenissen) mechanisme hebben, heeft Anydac componenten die toestaan te reageren op deze events.

Voor de mensen die geen behoefte hebben aan TDataset, heeft AnyDAC een eigen uniforme API die toestaat de databanken aan te spreken. Dat werkt iets sneller en heeft minder geheugen nodig.

AnyDAC is niet alleen beschikbaar voor Delphi, het kan ook met Free Pascal of Lazarus gebruikt worden. Uiteraard zijn op linux alleen databanken waarvoor Linux client libraries beschikbaar zijn, ondersteund.

In dit artikel tonen we hoe AnyDAC gebruikt kan worden om een Firebird databank aan te spreken; Maar de gebruikte technieken zijn dezelfde voor alle andere ondersteunde Databank protocollen.

2 Installatie

De installatie in Delphi is eenvoudig. De installer software installeert AnyDAC op windows, en zal een lijstje presenteren met ondersteunde databank types. Voor deze databank types kan een voorbeeld-databank opgevuld worden (de Northwind databank). Het is aan te bevelen tenminste 1 databank aan te maken, want alle voorbeeld code gaat ervan uit dat deze databank in een of andere vorm beschikbaar is. Welke RDBMS gebruikt wordt, is niet belangrijk, maar de client library moet beschikbaar zijn, en de databank moet reeds bestaan alvorens de installatie procedure gestart wordt.

De installatie zal ook aanbieden ondersteuning voor Synedit (syntax highlighting) te gebruiken in de GUI controls die SQL commandos tonen. Om hiervan gebruik te maken, moet Synedit reeds geinstalleerd zijn in de Delphi IDE. Als dit niet het geval is, zal de installatie van AnyDAC mislukken.

Nadat AnyDAC geinstalleerd is, zal de Delphi IDE 5 extra tabs hebben in de component palette:

AnyDAC Dit zijn de basis componenten om te verbinden met een databank, en queries uit te voeren: `TADConnection`, `TADManager` en `TADQuery` zijn de meest belangrijke componenten op deze tab.

AnyDAC UI Hier bevinden zich enkele componenten die op een form gezet moeten worden om enkele GUI mogelijkheden te activeren: een wacht cursor, een login dialoog en een complete visuele query builder dialoog.

AnyDAC links Deze tab bevat een link component voor elk ondersteund databank type: Minstens 1 van deze componenten moet op een form gezet worden om de connectie met een databank van dat type mogelijk te maken: het zorgt ervoor dat de driver code in de applicatie gelinkt wordt. De component staat ook toe enkele databank type specifieke opties in te stellen, zoals de naam van de client library. Door bijvoorbeeld een `TADPhysIBDriverLink` component op een form te zetten, zal Firebird support in het programma opgenomen worden.

AnyDAC Services Deze tab bevat componenten die databank-specifieke diensten aanbiedt, zoals backup en restore van een datank. Hier staan ook de event-notification componenten.

AnyDAC Devs Deze tab bevat enkele componenten voor algemene doeleinden, ze zijn strikt genomen niet nodig voor dagdagelijkse databank toegang.

De installatie bevat ook een 800 paginas tellende handleiding, zowel in Windows Help als in PDF formaat. De inleidende hoofdstukken zijn aanbevolen lectuur alvorens te beginnen programmeren met AnyDAC.

De AnyDAC installatie bevat niet alleen de componenten op de Delphi component palette, maar ook enkele programmas, die geinstalleerd worden in de `Bin` map van de installatie.

ADAdministrator Een administratie programma voor AnyDAC databank definities.

ADExplorer Een programma om SQL commandos uit te voeren op een van de door AnyDAC gekende databanken. Dit programma demonstreert veel van de AnyDAC mogelijkheden (de broncode is beschikbaar).

ADMonitor Een programma om de uitgevoerde SQL statements te bekijken. Het toont alle SQL commandos die door de AnyDAC componenten worden uitgevoerd.

Tenslotte zijn er een hoop voorbeeldprogrammas: elk deel van AnyDAC wordt gedemonstreerd in minstens 1 AnyDAC voorbeeld. De `Demos` project groep in de map Samples lijst alle voorbeelden op.

3 Architectuur

AnyDAC bestaat uit verschillende lagen die op elkaar voortborduren:

Physical layer De onderste laag is de 'physical layer' (fysische laag). Deze definieert een API voor data toegang, en bevat een implementatie van deze API voor alle gekende databank types. In wezen is dit de code die de RDBMS client library aanspreekt. Deze code voert de commandos uit, en wisselt de data uit tussen de Delphi applicatie en de client library. Deze laag wordt door de eindprogrammeur normaal niet gebruikt.

Adaptor layer Deze laag ligt boven de physical layer en presenteert een uniforme API voor databank toegang naar de programmeur. Deze laag kan op zich gebruikt worden, en bevat een hoop logica voor de creatie en optimalisatie van SQL commandos die door AnyDAC gegenereerd worden.

Component layer De componentt laag bevat de niet-visuele componenten die gebruikt worden in een applicatie. Bijvoorbeeld de connectie component `TADConnection` en query component `TADQuer` bevinden zich in deze laag.

GUI layer De GUI laag bevat enkele hulp componenten zoals de wachtcursor. Het zijn plug-ins voor de onderliggende lagen: De onderliggende laag kan zelf geen cursor te tonen, maar definieert een interface om een cursor te tonen. De onderliggende lagen weten niet hoe een login dialoog te tonen, maar definieren een interface voor het tonen van zo'n dialoog. De GUI laag bevat componenten die deze interfaces implementeren.

Er zijn nog enkele kleinere delen in AnyDAC, maar deze zijn zelden gebruikt. In de praktijk zal 1 component uit de physical layer gebruikt worden, en vele componenten uit de component laag.

4 Verbinding leggen met een databank.

Na alle theorie is het tijd AnyDAC in de praktijk te demonstreren door verbinding te maken met een databank.

Hiervoor wordt een klein programma gemaakt dat een databank met aanwezigheids gegevens aanspreekt: De databank bevat (fictieve) gegevens voor een school; Voor alle leerlingen in de school wordt bijgehouden wanneer zij de school binnenkomen en buitengaan. Er zijn 2 tabellen:

PUPIL Deze tabel bevat de namen van alle leerlingen in de school, in de velden `PU_FIRSTNAME` en `PU_LASTNAME`) en heeft een unieke sleutel `PU_ID` (een auto-incremental field)

PUPILTRACK Deze tabel bevat een record voor elke keer dat de leerling de school verlaat of binnenkomt. Het record bevat een type veld `PT_TYPE` met inhoud 'I' of 'O'

(voor 'in' of 'out'). Het bevat ook een tijdstip, en uiteraard een verwijzing naar een leerling.

De databank wordt gemaakt in Firebird, maar elk type RDBMs kan gebruikt worden.

Zoals de BDE aliases gebruikt, en ODBC gebruik maakt van DSN namen, werkt AnyDAC met een globaal connectie definitie bestand. Dit is een bestand waar alle door AnyDAC gekende databank verbindingen gedefinieerd zijn.

Het is een .ini bestand, met een sectie voor elke gekende databank verbinding. De sectie bevat de benodigde informatie om een driver voor de databank te selecteren, en de informatie die de driver nodig heeft om verbinding te leggen met de databank.

Het volgende is een voorbeeld van zulk een sectie. Het definieert een connectie 'Tracker':

```
[Tracker]
DriverID=IB
User_Name=MyUser
Password=MyPassword
Server=192.168.0.98
Database=/home/firebird/tracker.fdb
```

Het formaat is zeer eenvoudig; Elk installatieprogramma kan secties maken in dit bestand. Er zijn meer opties beschikbaar dan hier getoond worden. Zij kunnen eenvoudig beheerd worden op verschillende manieren:

1. Het `ADAdministrator` programma dat AnyDAC geinstalleerd heeft kan gebruikt worden om de definities te beheren. Dit programma is bedoeld voor de ontwikkelaar.
2. De `TADConnection` component editor kan de benodigde waardes in dit bestand schrijven. Dit mechanisme is enkel beschikbaar in de Delphi IDE.
3. De `TADManager` component kan de benodigde waardes in dit bestand beheren. De component leest en schrijft de nodige waardes. Alle opties die beschikbaar zijn, kunnen als properties van de component ingesteld worden. Het is dan ook bij uitstek de krachtigste oplossing.

Om verbinding te leggen met een databank in de Delphi IDE, moet een `TADConnection` component gebruikt worden. We geven er de naam `CTracker` aan. Een dubbelklik op de component toont de connectie-definitie dialoog, getoond in figuur 1 op pagina 5. Nadat de nodige parameters ingesteld zijn, kan de 'Test' knop gebruik worden om de connectie te testen. Indien de verbinding met success gemaakt werd, kan de 'Save' button gebruikt worden om de configuratie te bewaren, en de dialoog te sluiten.

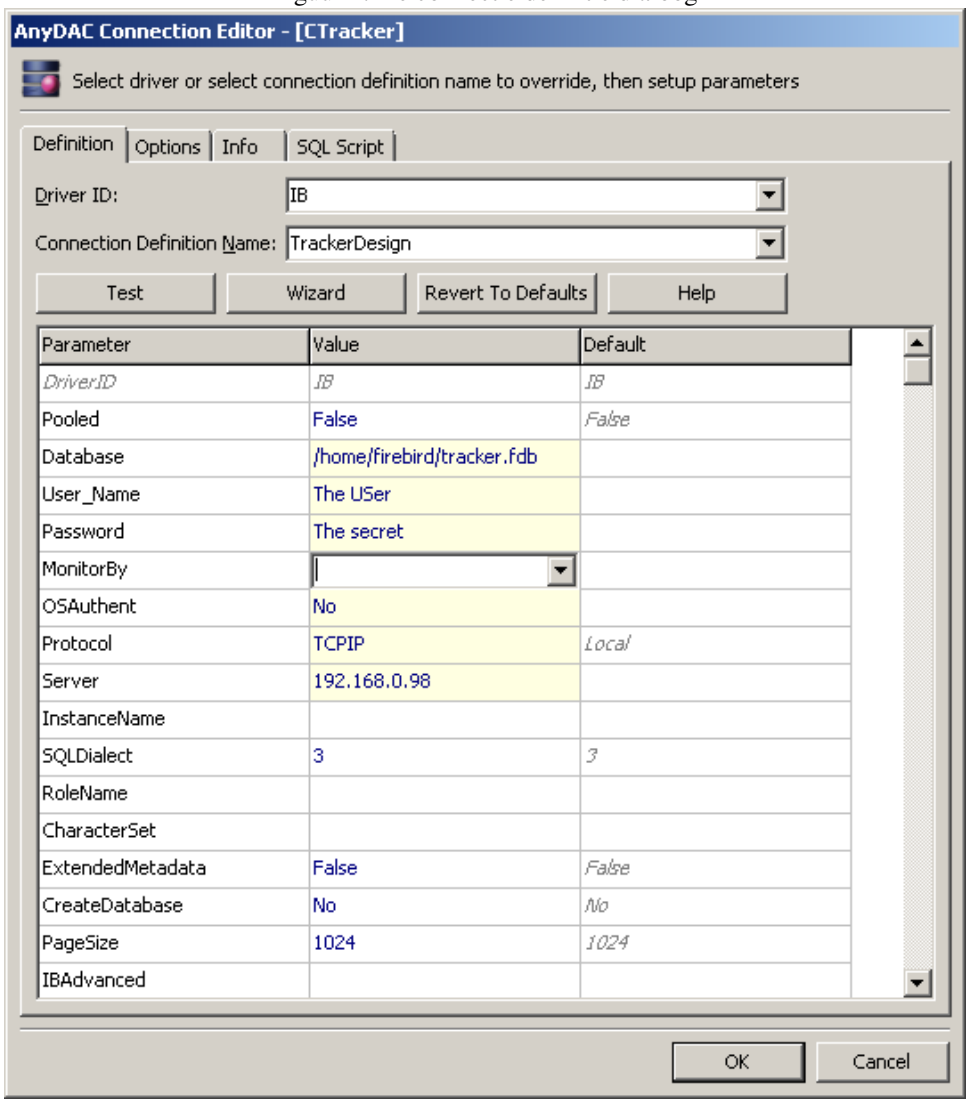
Door de `Connected` property van de component op 'True' te zetten, wordt de verbinding gemaakt. Deze stap is niet echt nodig: als een dataset, verbonden met de component, geopend wordt, zal de connectie automatisch geopend worden.

Indien de connectie reeds gedefinieerd was in het connectie bestand, dan volstaat het om de `ConnectionDefName` property van de `CTracker` component in te stellen op de naam van de connectie.

Runtime, op de computer waar het programma moet geinstalleerd worden, zal het connectie bestand allicht niet beschikbaar zijn, of de connectie is misschien niet gedefinieerd. De `TADManager` component kan dan gebruikt worden om de benodigde connectie aan te maken alvorens de verbinding met de databank te maken.

De tracker applicatie doet dit als volgt:

Figuur 1: De connectie definitie dialoog



```
procedure TForm1.CreateConnection;

Var
 I : IADStanConnectionDef;

begin
  I:=MGrAD.ConnectionDefs.FindConnectionDef('Tracker');
  if (I=Nil) then
    begin
    // create new connection
    I:=MGrAD.ConnectionDefs.AddConnectionDef;
    // Set the needed parameters
    I.DriverID:='IB';
    I.Name:='Tracker';
    I.UserName:='myuser';
    I.Password:='mypassword';
    I.Server:='192.168.0.98';
    I.Database:='/home/firebird/tracker.fdb';
    // Optional.
    I.MarkPersistent;
    I.Apply;
    end;
end;
```

`MgrAD` is een `TADManager` component instance. De eerste lijn kijkt na of een `Tracker` connectie definitie bestaat. Als er geen bestaat, wordt er een aangemaakt. De properties spreken voor zich, ze behoeven geen verdere uitleg.

De laatste 2 lijnen zijn optioneel: De definitie kan tijdelijk zijn. In dat geval wordt ze niet bewaard in het connectie bestand, maar verdwijnt zodra het programma afgesloten wordt. Om de definitie toch te bewaren, moet de `MarkPersistent` methode opgeroepen worden. Deze vertelt de `TADManager` component dat de definitie in het connectiebestand bewaard moet worden. De oproep van `Apply` zorgt er voor dat de definite daadwerkelijk geschreven wordt.

Nadat de connectie gedefinieerd is, kan de verbinding gelegd worden:

```
procedure TForm1.Connect;

begin
  CTracker.ConnectionDefName:='Tracker';
  CTracker.Connected:=True;
end;
```

De eerste lijn stelt de net gedefinieerde connectie verbinding in. De tweede lijn opent de verbinding.

5 Fetching data from the database

Nu het programma verbonden is met de databank, kan data opgehaald worden. De bedoeling is een lijst leerlingen te tonen in een grid. Zodra een leerling geselecteerd wordt, zullen de aanwezigheidsgegevens in een 2de grid getoond worden.

Om gegevens uit een databank op te halen, biedt AnyDAC 3 componenten aan:

TADQuery Een `TDataset` descendant waarin een SQL SELECT commando kan opgegeven worden.

TADTable Een `TDataset` descendant waarbij een tabelnaam kan opgegeven worden. Alle data van deze tabel wordt opgehaald.

TADStoredproc Een `TDataset` descendant waarbij de naam van een stored procedure kan opgegeven worden, samen met de waardes voor de parameters die de procedure nodig heeft. De stored procedure wordt uitgevoerd, en alle data die de procedure teruglevert wordt opgehaald.

Deze componenten hebben dezelfde functie als hun equivalenten in DBX, BDE of ADO technologieen.

Data kan met behulp van een van deze componenten opgehaald worden zoals dat in alle andere Delphi data componenten gebeurt: Een `TADQuery` component wordt op de form geplaatst (`QPupils`), en de `Connection` property wordt automatisch ingesteld op de `CTracker` component. De `SQL` property kan eenvoudig ingesteld worden op:

```
SELECT * FROM PUPIL;
```

De `Active` property op `True` zetten, zal de dataset openen. Dan wordt een `TDatasource` component (naam: `DSPupils`) met de dataset verbonden, en met een `TDBGrid` control verbonden. Dit toont de lijst van leerlingen op het scherm, in de IDE.

6 Using Parameters to create master/detail

Nu de leerlingen op het scherm staan, kunnen de aanwezigheidsgegevens van een geselecteerde leerling getoond worden. Dit is een typische Master/Detail relatie tussen datasets. Zoals te verwachten in een Delphi data technologie, geeft AnyDAC de mogelijkheid queries te parametriseren, en dit te gebruiken in een master/detail relatie tussen datasets.

Parameters kunnen in een `TADQuery` component opgegeven worden op dezelfde manier als in de andere beschikbare technologieen. Om dit te tonen, plaatsen we een tweede `TADQuery` component (naam: `QTrack`) op de form. Dubbelklikken op de component toont de SQL editor. Deze editor staat toe op een visuele manier een SQL Select commando op te bouwen. Dit wordt getoond in figuur 2 op pagina 8. Met deze editor kan het volgende SQL commando gemaakt worden:

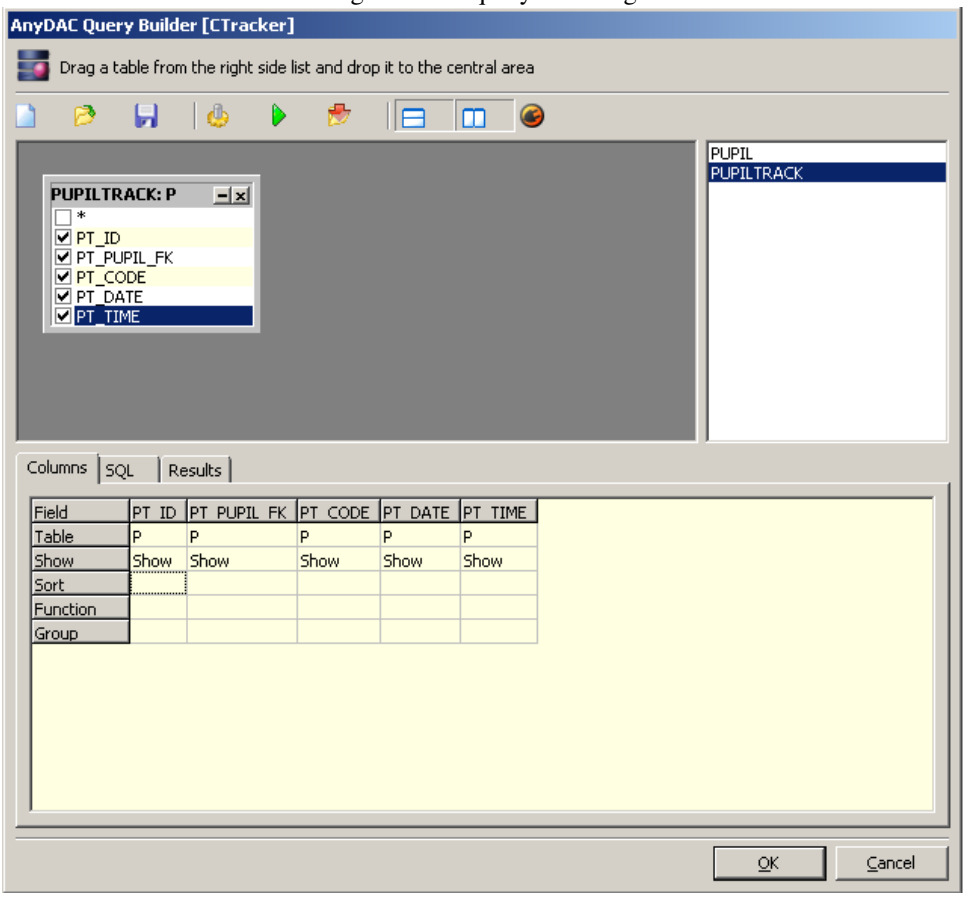
```
SELECT
  P.PT_ID, P.PT_PUPIL_FK, P.PT_CODE, P.PT_DATE, P.PT_TIME
FROM
  PUPILTRACK P
WHERE
  (PT_PUPIL_FK=:PU_ID)
```

Merk op dat de editor automatisch een tabel alias aanmaakt voor alle tabellen (in dit geval: P). Een parameter wordt in het SQL commando opgegeven op de gebruikelijke manier: `:PU_ID`. De waarde kan in code gezet worden als volgt:

```
  QTrack.Params.ParamByName('PU_ID').AsInteger:=1;
```

Om een echte master-detail relatie te maken kan de `MasterSource` property van `QTrack` ingesteld worden op `DSPupils`. Nu, wanneer `QTrack` geopend wordt, zal de waarde van de `PU_ID` parameter opgehaald worden uit het huidige record in `QPupils`. Bovendien

Figuur 2: De query dialoÃăg



zal, wanneer er in de QPupils data genavigeerd wordt, de QTrack query de data opnieuw ophalen zodra het huidige record in QPupils wijzigt.

AnyDAC ondersteunt niet alleen parameters, het ondersteunt ook macros. Parameters kunnen gebruikt worden om hetzelfde SQL commando meerdere keren uit te voeren, met telkens een andere waarde voor de parameter. Als een RDBMs het gebruik van parameters ondersteunt, zal AnyDAC hiervan gebruik maken, hetgeen de efficientie verhoogt. Macros daarentegen kunnen gebruikt worden om het SQL commando zelf aan te passen: Macros worden door hun waarde vervangen voordat het SQL commando aan de RDBMs doorgegeven wordt. In AnyDac worden macros aangeduid dmv. de ! of & karakters.

Om het gebruik van macros te demonstreren, brengen we een kleine feature aan in de tracker: Een klik op de hoofding van een kolom in de grid met leerlingen, sorteert de lijst van leerlingen op het veld, getoond in die kolom. Een tweede maal klikken, keert de sorteervolgorde om.

Om dit te doen wordt de SQL property van QPupils ingesteld op:

```
SELECT * FROM PUPIL ORDER BY !SORTFIELD !SORTORDER
```

De query heeft nu 2 macros: SORTFIELD en SORTORDER.

De OnTitleClick event van de grid met de leerlingen kan als volgt gecodeerd worden:

```
procedure TForm1.DBGrid1TitleClick(Column: TColumn);

Var
  FN : String;
  MFN : TADMacro;
  MSO : TADMacro;

begin
  FN:=Column.FieldName;
  QPupils.Close;
  MFN:=QPupils.Macros.MacroByName('SORTFIELD');
  MSO:=QPupils.Macros.MacroByName('SORTORDER');
  if (FN=MFN.AsIdentifier) then
    begin
    // Same field, reverse sort order
    If MSO.AsRaw='ASC' then
      MSO.AsRaw:='DESC'
    else
      MSO.AsRaw:='ASC';
    end
  else
    begin
    // Other field. Set sort order to ASC
    MFN.Asidentifier:=FN;
    MSO.AsRaw:='ASC';
    end;
  QPupils.Open;
end;
```

Dit is vrij eenvoudige code: de query wordt gesloten, en de waarden voor de macros worden in lokale variabelen opgeslagen. Indien de gebruiker op een andere kolomhoofding heeft geklikt, wordt de nieuwe waarde van de sortfield macro ingesteld. Indien dezelfde

kolomhoofding aangeklikt werd, wordt de sorteervolgorde omgedraaid. Daarna wordt de query weer geopend.

Merk het gebruik op van de properties `AsRaw` en `AsIdentifier`: De macros ondersteunen niet alleen rechttoe-rechtaan tekstvervanging, maar ze begrijpen ook SQL: Aan elke macro is een data type geassocieerd: Voor `SortField` is dit `mdIdentifier`, omdat de macro een identifier bevat. `AsIdentifier` instellen zal de waarde van de macro met quote karakters omringen indien dit vereist is. Het type `mdRaw` betekent dat gewoon tekst substitutie verricht wordt. De waarde moet dan d.m.v. de `AsRaw` property gezet worden.

Als niet alle records van de server opgehaald worden (dmv. buffering), is dit de eenvoudigste manier om de records in de grid te sorteren. Indien alle records opgehaald worden, is het eenvoudiger en sneller de ondersteuning voor lokale indexen van `TADQuery` te gebruiken.

7 Data bewerken

AnyDAC ondersteunt niet alleen het tonen, maar ook het bewerken van data in een dataset. De `UpdateActions` property van `TADQuery` geeft enkele opties om de toegestane operaties (insert/update/delete) aan of uit te schakelen. Er zijn ook properties om aan te geven hoe deze operaties dienen te gebeuren. AnyDAC genereert dan de nodige update SQL commandos om de gegevens in de databank bij te werken. Om deze commandos te kunnen aanmaken, moet AnyDAC een unieke sleutel hebben voor een record: de namen van de velden die de unieke sleutel vormen, moeten worden opgegeven. Dit kan op 3 verschillende wijzen:

1. De standaard manier is de informatie over de unieke sleutel op te halen in de databank. Dit gebeurt indien `fiMeta` deel uitmaakt van de `fetchOptions.Items` property van de `TADQuery` component. Indien deze optie gezet is, wordt de naam van de eerste tabel in de FROM clausule van het SQL SELECT commando genomen, en de unieke sleutel van die tabel wordt opgevraagd.

   Het nadeel van dit mechanisme is dat het extra queries vereist wanneer de dataset open gaat, en dus trager werkt.

2. Bij gebruik van persistent fields kan de `pfInKey` optie in de `ProviderOptions` van de sleutelvelden gezet worden.

3. De namen van de sleutelvelden kunnen opgegeven worden in de `UpdateOptions.KeyFields` property (gescheiden door puntcommas). Als deze property gezet is, wordt de `pfInKey` optie in de `ProviderOptions` property genegeerd.

De `fiMeta` optie is standaard geactiveerd. Ze moet verwijderd worden uit de `fetchOptions.Items` property om een van de laatste 2 opties te kunnen gebruiken.

De `UpdateOptions` property verschaft nog meer opties om de update operaties te beinvloeden. Als de standaard mechanismen van AnyDAC niet volstaan, kan een `UpdateObject` opgegeven worden: In dit object kunnen de SQL statements voor alle update operaties ingesteld worden. Een dubbelklik op de component toont een dialoog die initiele versies voor de nodige statements aanmaakt.

8 Import/Export

Naast eenvoudige data toegang, heeft AnyDAC nog enkele andere nuttige componenten. Een ervan is de `TADDataMove` component. Deze kan gebruikt worden om data te migre-

ren tussen 2 datasets of tussen een CSV bestand en een dataset. Het enige dat gedaan dient te worden is bron en doel voor de data op te geven. Dit kan gebruikt worden om snel en eenvoudig data in een databank te laden, of om data te exporteren uit een databank.

De lijst van leerlingen kan bijvoorbeeld op eenvoudige wijze geexporteerd worden. Om dit te doen volstaat het een `TADDataMove` component op de form te zetten (naam: DMPupils), en als `Source` property `QPupils` in te stellen. De `DestinationKind` property moet op `skText` gezet worden. Om de gebruiker een bestandsnaam te laten opgeven, gebruiken we een `TSaveFileDialog` component (`SDDump`). De eigenlijke export kan dan als volgt gecodeerd worden

```
procedure TForm1.ADumpPupilsExecute(Sender: TObject);

Var
  FN : String;
begin
  if not SDDump.execute then
    exit
  else
    FN:=SDDump.FileName;
  FADGUIxSilentMode:=True;
  With DMPupils do
    begin
    TextFileName:=FN;
    Execute;
    end;
end;
```

De code is weerom heel eenvoudig. De `TextFileName` property wordt ingesteld en de `Execute` methode doet al het werk om het CSV bestand aan te maken.

De data move component kan echter meer: het kan data lezen van het CSV bestand, en dit aan de dataset toevoegen. Het enige dat hiervoor vereist is, is de bron en doel properties omkeren. Het voorbeeld programma bevat code dat toont hoe dit moet.

Indien niet alle velden nodig zijn, kan de `Mappings` property gebruikt worden om aan te geven welke velden moeten geexporteerd worden. De component kan ook een logbestand aannmaken. Dit is bijzonder nuttig om een tekstbestand in te lezen, in combinatie met de mogelijkheid om exceptions te negeren (bijvoorbeeld wanneer eenzelfde unieke sleutel tweemaal wordt ingelezen).

9 Conclusie

AnyDAC is een uitgebreide reeks componenten voor data toegang. Het kan zonder twijfel elke bestaande data technologie binnen Delphi vervangen. In dit artikel werd slechts een klein deel van de mogelijkheden van AnyDAC getoond. Services, batch operaties, alarmen, monitoring, scripts en conditioneel afhandelen van SQL commandos, lokale datasets zijn slechts enkele van de mogelijkheden die niet behandeld werden. Zij zullen in een volgende bijdrage behandeld worden.