Access Voor Beginners - Hoofdstuk 11 (deel 2)

Handleiding van Helpmij.nl

Auteur: OctaFish

April 2012

In het vorige hoofdstuk hebben we ons bezig gehouden met rapporten maken via de wizard en hoe een rapport is opgebouwd. Meestal zullen we rapporten maken die altijd op één manier zullen worden gevuld. Denk bijvoorbeeld aan een factuur voor een klant, die altijd een standaard lay-out zal hebben, met alleen de klant- en bestelgegevens die veranderen. Ook overzichten kun je op die manier maken, waarbij je voor verschillende soorten overzichten aparte rapporten maakt. Je kunt bijvoorbeeld een vast rapport maken voor de maandoverzichten, weekoverzichten etc.

In dit hoofdstuk gaan we een echter een stapje verder, en maken we een interactief rapport waarbij we met behulp van keuzelijsten eerst een selectie maken, en op basis daarvan een rapport laten vullen. We gebruiken daarvoor de meest ingewikkelde variant: een kruistabel. Iedereen die wel eens een kruistabel heeft gemaakt, weet dat de kolomkoppen in een kruistabel dynamisch zijn: ze worden afhankelijk van de beschikbare gegevens aangemaakt. In een rapport op basis van die kruistabel zie je echter niet de veldnaam terug voor de kolomkoppen, maar de feitelijke gegevens die in de kruistabel zijn gemaakt voor de kolomkoppen.

Als je bijvoorbeeld de maandomzetten van vorig jaar in een kruistabel zet, zul je vermoedelijk voor elke maand wel gegevens hebben. Gebruik je de naam van de maand als kolomkop, dan zul je dus 12 kolommen krijgen met gegevens. Maak je nu een rapport op basis van die kruistabel, dan krijg je in het rapport 12 kolommen terug die namen hebben als: januari, februari etc. Het veld [Maandnaam] uit de query is dus vervangen door 12 velden met de namen van de 12 maanden.

Wil je dezelfde kruistabel maken voor het huidige jaar, dan zie je alleen maar gegevens voor de maanden waarvoor je gegevens hebt, en die lopen dus logischerwijs tot en met de huidige maand. Voor de resterende maanden zie je dus geen kolommen in de kruistabel. Maak je het rapport op basis van de gegevens van het huidige jaar, dan mis je in het rapport dus de noodzakelijke kolommen. In het rapport dat je voor het vorige jaar had gemaakt, heb je voor die ontbrekende maanden wel velden staan, die dus een foutmelding opleveren als je het rapport afdrukt: de velden van de ontbrekende namen zitten namelijk niet in de query.

Daar is nog wel iets op te verzinnen, maar daar gaat dit hoofdstuk niet over. En wat nu, als je hetzelfde rapport wilt gebruiken, maar nu op voor de Artikelcategorieën? Je kunt dan een nieuw rapport maken en vervolgens alle velden vervangen, maar dat is een hoop werk. Zou het niet makkelijker zijn als je alle gegevens die je in de kruistabel wilt gebruiken, vrij kunt kiezen?

In dit hoofdstuk gaan we een variabel rapport maken op basis van omzetcijfers. Deze omzetcijfers zijn gebaseerd op een tabel Artikelen, en een tabel met verkoopgegevens. We willen een rapport kunnen maken op basis van de verkochte artikelen, en op basis van de bij die artikelen behorende artikelgroepen. Tevens willen we een rapport kunnen maken op basis van een datum, of een aantal verschillende datums.

Opdracht

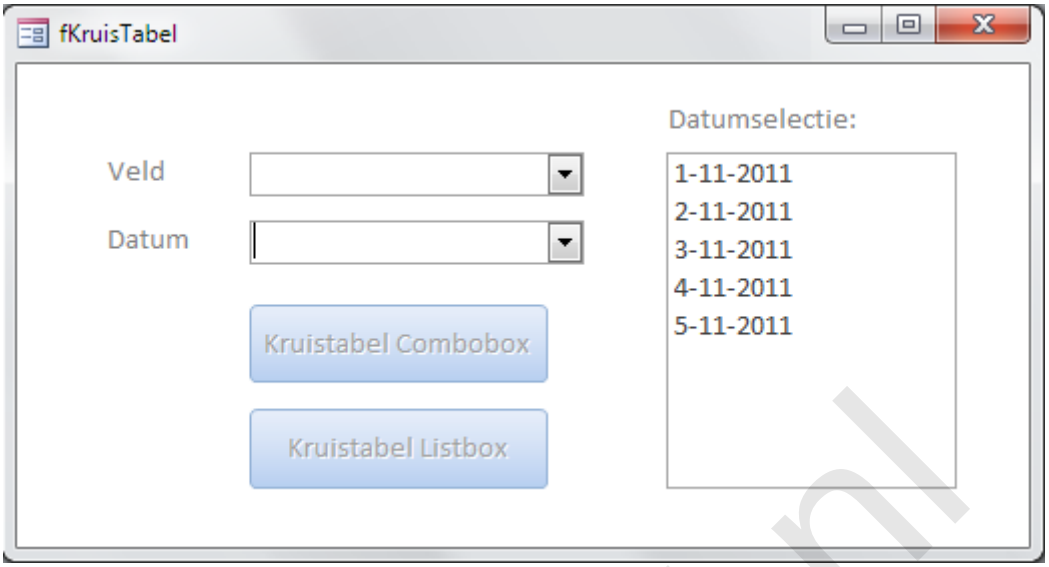
Maak een nieuwe query, en sla deze op onder de naam **qTemp**. Dit mag een selectiequery zijn, een Kruistabelquery of welk querytype dan ook. Deze query wordt later gebruikt als Kruistabelquery voor het rapport.

Stap 1: het selectieformulier

Omdat een rapport gekoppeld is aan een gegevensbron, en we het rapport in beginsel niet steeds aan een andere query willen koppelen, kiezen we een vaste query als basis voor het rapport. Op die manier hoeven we het rapport zelf niet steeds aan een nieuwe bron te koppelen, wat alleen in de Design modus kan. Er zijn echter wel een paar regels die we moeten volgen:

1. Het rapport wordt gekoppeld aan een vaste recordbron, een query.
2. In het rapport worden standaardvelden gemaakt, die bij het openen van het rapport worden gekoppeld aan de wisselende velden.
3. De query wordt aangepast op basis van keuzes die in een formulier worden gemaakt.

Het formulier ziet er zo uit:



Dit formulier bevat zoals je ziet twee keuzelijsten met Invoervak, en één keuzelijst met datums. De keuzelijst Veld is gekoppeld aan de tabel Artikelen, en laat een lijst met velden zien. In deze keuzelijst kunnen we het veld kiezen waarop we de gegevens willen totaliseren. Je zou ook kunnen zeggen dat de keuzelijst <Veld> straks de kolomkoppen levert voor de kruistabel, en dus ook voor het rapport.

De keuzelijsten Datum en Datumselectie kun je gebruiken om ofwel één datum te selecteren, ofwel een aantal beschikbare datums te selecteren.

Opmerking: als je één datum wilt selecteren, kun je natuurlijk ook de keuzelijst gebruiken. Wil je echter een datumreeks maken, dan zijn twee keuzelijsten met invoervak aan te bevelen, omdat je daarmee maar één datum kunt selecteren, en je bij een keuzelijst een keuze maakt voor ofwel één datum (gelijk aan een keuzelijst met invoervak) ofwel een niet-gelimiteerd aantal waarden. En die laatste instelling maakt een keuzelijst dus minder geschikt voor het vastleggen van een datumreeks die altijd tussen twee datums moet liggen, want je kunt niet afdwingen dat er maar twee waarden mogen worden gekozen.

In deze oefening maken we dus een rapport waarbij we ofwel één waarde kiezen via de Keuzelijst met invoervak, ofwel verschillende datums die we dan met de keuzelijst kiezen.

Zoals je in de afbeelding kunt zien, zijn er twee knoppen gemaakt waarmee we het rapport maken en openen. Je zou alle code, als je dat zou willen, onder één knop kunnen maken. Om de oefening niet te ingewikkeld te maken, houden we het hier op twee aparte knoppen: één knop die de waarde uit de Keuzelijst met invoervak gebruikt, en één knop die de waarden uit de Keuzelijst uitleest.

Op de twee keuzelijsten met Invoervak zit een check die voorkomt dat de knop cmdKruistabel wordt aangeklikt; zoals je in het plaatje kunt zien, is deze knop uitgeschakeld om te voorkomen dat het rapport wordt geopend zonder dat er een veld en een datum is geselecteerd. Hetzelfde geldt voor de knop cmdKruistabel_List, die minstens één datum nodig heeft, en een veldnaam uit de keuzelijst Veldnaam.

De keuzelijsten zijn als volgt opgebouwd:

Keuzelijst met invoervak <cboDatum>

Type Rijbron: Tabel/Query
Rijbron: Tabel, of Query met daarin het veld [Datum]
Gebeurtenis: Na bijwerken; code om de knop cmdKruistabel te activeren.

Codevoorbeeld:

```
Private Sub cboDatum_AfterUpdate()

   If Not Me.cboDatum = "" And Not Me.cboVeld = "" Then

      Me.cmdKruistabel.Enabled = True

   End If

End Sub
```

De code is op zich vrij eenvoudig te begrijpen, als je de eerdere hoofdstukken hebt gevolgd. Er wordt een IF-functie gebruikt die kijkt of de keuzelijst met Invoervak cboDatum ingevuld is, en of er een veld is gekozen in de keuzelijst met invoervak cboVeld. Als aan die voorwaarden is voldaan, wordt de knop cmdKruistabel geactiveerd.

Keuzelijst met invoervak <lstDatum>

Type Rijbron: Tabel/Query
Rijbron: Tabel, of Query met daarin het veld [Datum]
Meervoudige Selectie: < Enkelvoudig> of < Uitgebreid>
Gebeurtenis: Na bijwerken; code om de knop cmdKruistabel_Lijst te activeren.

Keuzelijst met invoervak <cboVeld>

Type Rijbron: Lijst met velden

Rijbron: Artikel

Gebeurtenis: Na bijwerken, code om ofwel de knop cmdKruistabel_Lijst te activeren als er een datum (of meer) uit de keuzelijst wordt gekozen, ofwel de knop cmdKruistabel als de keuzelijst met Invoervak wordt gebruikt.

Opdracht

- Maak het formulier fKruistabel, met daarop minstens twee Keuzelijsten met Invoervak, waarvan één keuzelijst voor het kiezen van velden uit de tabel Artikelen, en één keuzelijst voor het selecteren van een Datum uit de gegevenstabel.
- Maak een Keuzelijst met dezelfde gegevensbron als de keuzelijst cboDatum.
- Maak de code die de juiste knop activeert bij de keuzelijsten

Een dynamische query maken

Bij een dynamisch rapport, hoort een dynamische recordbron. Daarvoor kunnen we in beginsel elke query gebruiken die we daarvoor aanwijzen; wat er straks gaat gebeuren met de twee opdrachtknoppen, is dat er een SQL-string wordt gemaakt, die daarna a.h.w. in een bestaande query wordt geplakt. Het effect daarvan is, dat die query dus ineens een hele andere query is. Het maakt dan ook niet uit of je een query maakt die een veld uit een tabel selecteert, of een Delete query, of een Tabelmaak query: zodra we de knop activeren, wordt de complete SQL van die query vervangen door de nieuwe SQL die we maken op basis van onze selectie! Met deze techniek kun je dus met één query volstaan!

De daarvoor benodigde code bevindt zich dus onder de knoppen < cmdKruistabel > en <cmdKruistabel_List>, en daar gaan we nu naar kijken. Eerst de code voor de Keuzelijst met Invoervak, want die is het eenvoudigst.

Zoals gebruikelijk, beginnen we eerst weer met het declareren van de variabelen die we nodig hebben. In dit geval zijn dat de volgende variabelen:

```
Dim qTmp As QueryDef      Object'Variabele van het type Query Definitie
Dim lst As ListBox        'ObjectVariabele van het type Listbox
Dim itm As Variant        'Variabele van het type Variant

Dim iDatum As Long        'GetalVariabele van het type Long

Dim dtDatum As Date       'GetalVariabele van het type Date
```

En de procedure die we maken is een <Bij klikken> gebeurtenis op de knop Kruistabel.

```
Private Sub cmdKruistabel_Click()

'Eerst de kruistabel query maken op basis van de geselecteerde datum

dtDatum = Me.cboDatum

iDatum = CLng(dtDatum)
```

De eerste stap die we doen is het omzetten van de geselecteerde datum naar een Getal. Dat lijkt vreemd, en is ook niet altijd nodig, maar omdat in VBA/SQL de Amerikaanse Datumnotatie wordt gehanteerd willen we voorkomen dat de datum 4-9-2012 (4 september dus) wordt gelezen als 9-4-2012 (wat 9 april is). Om die omwisseling van Maand en Dag te voorkomen, wordt de datum dus eerst vertaald naar de feitelijke getalswaarde. In de query zie je straks dan ook de opdracht **CDate(" & iDatum & "))** terugkomen, en met CDate wordt het getal weer terugvertaald naar een datum.

```
strPivot = "TRANSFORM Sum(Tabel.Aantal) AS Totaal " & vbCrLf _

   & "SELECT Datum " & vbCrLf _

   & "FROM Artikel INNER JOIN Tabel ON " _

   & "Artikel.ArtikelID = Tabel.Artikel " & vbCrLf _

   & "WHERE (Datum = CDate(" & iDatum & ")) " & vbCrLf _

   & "GROUP BY Datum " & vbCrLf _

   & "ORDER BY " & Me.cboVeld & " " & vbCrLf _

   & "PIVOT Artikelnaam;"
```

De code maakt een Kruistabel query aan. Deze query variant herken je aan het beginwoord TRANSFORM. Een selectiequery begint altijd met SELECT. Een kruistabel heeft altijd drie elementen: een Rijkop, een Kolomkop en een Waardeveld. In SQL herken je die niet 1-2-3, maar met een beetje puzzelen zijn ze wel terug te vinden. Zo is het Waardeveld altijd een Berekening. In de code zie je één berekening staan: **Sum(Tabel.Aantal) AS Totaal**. Dit zal dus wel het Waardeveld zijn! Een Kruistabel heeft ook minstens één Rijkop (je kunt meerdere rijkoppen gebruiken, mocht je dat willen). Die Rijkop heeft altijd de functie <Group By>, en in deze code vind je die dan ook terug, namelijk bij: **GROUP BY Datum**. Het veld [Datum] zal dus wel een Rijkop zijn. Dat laat alleen nog het veld over dat wordt gebruikt voor de Kolommen, en waarvoor je de waarden wilt berekenen. In bovenstaand voorbeeld is dat het veld [Artikelnaam], en je vindt die terug in de regel **PIVOT Artikelnaam**.

Het volledig intypen van een Kruistabelquery is dus een redelijk lastige zaak, want je hebt met een aantal zaken te maken die van zichzelf niet helemaal logisch zijn of lijken, iets wat bij een SELECT query wèl het geval is: de query SELECT [Artikelnaam, Datum * FROM Tabel is snel gemaakt.

In het voorbeeld dat we gaan maken, is het zelfs nog iets ingewikkelder, omdat we het PIVOT veld variabel willen maken: daar hebben we tenslotte een Keuzelijst met Invoervak voor. De laatste regel wordt dan ook in de uiteindelijke query zo:

```
& "PIVOT " & Me.cboVeld & ";"
```

Opmerking: niet alle velden in deze keuzelijst zijn geschikt om te gebruiken voor de kruistabel. Wil je een keuzelijst maken die daar rekening mee houdt, dan zou je een aparte tabel kunnen aanleggen met veldnamen die je daarvoor mag gebruiken.

De volgend stap is, om de Pivot query als SQL bron toe te wijzen aan de vaste query die we daarvoor gemaakt hebben. In het voorbeeld heb ik die query qTemp genoemd. We doen dat middels de variabele qTmp die als QueryDef is gedeclareerd. In de volgende opdracht wordt de query qTemp toegekend aan de variabele, en vervolgens krijgt deze de nieuwe SQL code toegewezen.

```
Set qTmp = CurrentDb.QueryDefs("qTemp")

qTmp.SQL = strPivot
```

Code voor de knop cmdKruistabel_List

De code voor de listbox knop is een beetje uitgebreider, omdat we nu moeten controleren hoeveel datums er zijn geselecteerd in de listbox. Dat doen we met een lus, die door alle geselecteerde items van de listbox loopt. Op basis van die keuzes wordt een filterstring gemaakt.

```
Set lst = Me.lstDatum
sFilter = "WHERE "
```

We beginnen met de keuzelijst aan de Listbox variabele lst toe te wijzen. In de tweede regel wordt de begintekst van het filter ingesteld. In de volgende regel begint de lus, waarbij elk geselecteerd item uit de list wordt toegewezen aan de variabele itm.

```
For Each itm In lst.ItemsSelected
   tmp = lst.ItemData(itm)
```

Omdat ook hier een datum is geselecteerd uit de lijst, wordt de datum op dezelfde manier als bij de keuzelijst met invoervak omgezet naar een getal.

```
   dtDatum = lst.ItemData(itm)

   iDatum = CLng(dtDatum)

   sFilter = sFilter & "(Datum = CDate(" & iDatum & ")) "
```

Vervolgens wordt dat getal aan het filter toegevoegd met de functie CDate, zodat we in de query weer datums terug krijgen. In de volgende stappen kijken we of we al bij het laatste geselecteerde item zijn of niet. Als dat niet zo is, dan moet het filter worden uitgebreid met het woord OR. Dat is dus al het geval als er twee datums zijn geselecteerd. Als laatste wordt de teller met de waarde 1 verhoogd.

```
   If i < lst.ItemsSelected.Count - 1 Then

      sFilter = sFilter & "OR "
```

```
    End If

    i = i + 1

Next itm
```

Als laatste moet de variabele strPivot nog worden aangepast in:

```
    & "Artikel.ArtikelID = Tabel.Artikel " & vbCrLf _

    & sFilter & vbCrLf _

    & "GROUP BY Datum " & vbCrLf _
```

Want het filter hebben we nu in een variabele gezet.

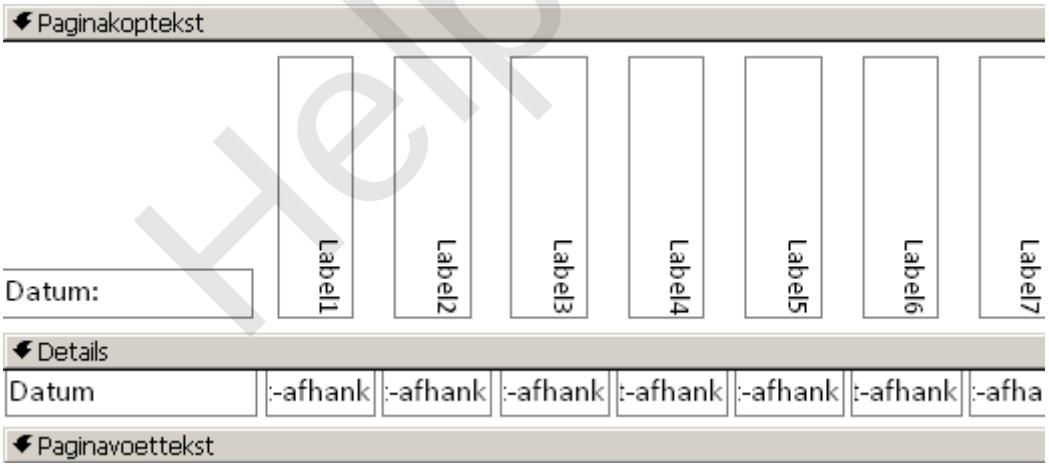
Opdracht

- Maak de code voor de knop cmdKruistabel en voor de knop cmdKruistabel_List.

De opbouw van het rapport

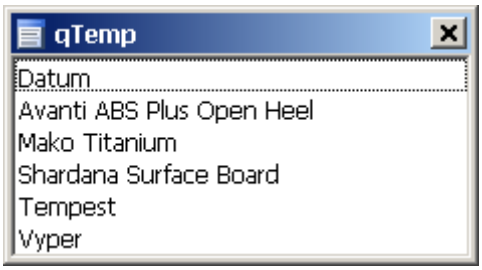
In dit voorbeeld wordt de query nogal variabel gemaakt, met allerlei verschillende velden. Hoe krijgen we die velden nu te zien op het rapport? Daarvoor moeten we naar het ontwerp van het rapport gaan kijken, en naar de procedures die daar plaatsvinden.

Het ontwerpscherm van het rapport ziet er zo uit:



In totaal zijn er 15 niet-afhankelijke tekstvakken die intrinsiek dezelfde naam hebben: v1 t/m v15, en 15 labels met de namen l1 t/m l15. Links van de onafhankelijke tekstvakken staat het Datumveld dat in elk rapport terugkomt. De 15 tekstvakken en labels hebben de eigenschap <Zichtbaar> = Nee, zodat zowel de tekstvakken als de labels in eerste instantie niet getoond worden in het rapport.

Het rapport is via de eigenschap <Recordbron> gekoppeld aan de query qTemp. Als je het venster <Lijst met velden> opent, zie je dus de velden die op dat moment in de query zijn opgenomen. Dat ziet er bijvoorbeeld dan zo uit:



Normaal gesproken zou je nu alle velden (behalve het datumveld, want dat is al gekoppeld) aan een van de niet-afhankelijke tekstvakken kunnen koppelen, en de eigenschap <Zichtbaar> op <Ja> instellen, zodat je de tekstvelden ook nog kan zien. En dat is exact wat we gaan doen, alleen doen we dat met een VBA-procedure, en niet met de hand. Dat gaat niet alleen sneller, het gaat ook nog eens zonder mogelijke fouten (als de code natuurlijk goed is J)

Opdracht

- Maak een nieuw rapport
- Koppel het rapport aan de query qTemp
- Zet het veld [Datum] als eerste veld op het rapport
- Maak daarin 15 niet-afhankelijke testvelden en geef ze identieke namen met een volgnummer, en doe dit ook voor de labels
- Maak alle niet-afhankelijke tekstvakken en bijbehorende labels onzichtbaar

Om de tekstvakken zichtbaar te maken, gebruiken we de volgende routine die wordt uitgevoerd bij het openen van het rapport.

```
Private Sub Report_Open(Cancel As Integer)

Dim sVelden() As String

Dim i As Integer, j As Integer
```

We beginnen met het uitlezen van de huidige veldnamen in de query qTemp. Deze veldnamen hebben we nodig om de Labels te vullen. We openen dus eerst een Recordset, en maken een lus die door alle velden loopt. Van die velden gebruiken we de eigenschap Name. De lust begint op 1 wat logisch lijkt, maar in dit geval is dat het tweede veld, omdat de lus bij 0 begint. Het Field(0) is dus het eerste veld in de tabel. In ons geval is dat het veld Datum, en dat staat al in het rapport. Deze constructie is ook de reden dat we het aantal velden op deze manier berekenen:

```
i = .Fields.Count-1
```

We gebruiken voor het opslaan van de veldnamen een matrix variabele,waarvan de grootte wordt ingesteld op het aantal velden.

```
With CurrentDb.OpenRecordset("SELECT * FROM qTemp")

   i = .Fields.Count-1

   For i = 1 To i

      ReDim Preserve sVelden(j)
```

Vervolgens gaan we de variabele vullen met de veldnamen.

```
      sVelden(j) = .Fields(i).Name
```

```
        j = j + 1
    Next i
End With
```

Als we alle veldnamen hebben, kunnen we ze koppelen aan de verschillende tekstvelden in het rapport. En de veldnaam kunnen we mooi gebruiken als Caption voor de labels, dus dat pakken we ook mee. En als laatste handeling maken we de velden ook zichtbaar, anders is alles alsnog voor niks geweest!

```
On Error Resume Next
For i = LBound(sVelden) To UBound(sVelden)
    Me("v" & i + 1).ControlSource = sVelden(i)
    Me("l" & i + 1).Caption = sVelden(i)
    Me("v" & i + 1).Visible = True
    Me("l" & i + 1).Visible = True
Next i
End Sub
```

Je kunt in VBA met het gereserveerde word **Me.** verwijzen naar een object op formulier of rapport. In dit geval gaat dat niet, omdat de objectnamen gegenereerd worden op basis tekst en een nummer. We zullen dus op een iets andere manier naar de objecten moeten verwijzen. In dit geval doen we dat door de objectnaam uit de *Collectie* van ME te halen. Daarbij kun je verwijzen naar de *indexnaam* van het object, of naar het *indexnummer* van het object. Aangezien we het indexnummer niet weten, gebruiken we de indexnaam. In essentie zijn deze vier commando's dus gelijk als i de waarde 5 heeft:

```
Me.v6.ControlSource = sVelden(i)
Me("v6").ControlSource = sVelden(i)
Me(v6).ControlSource = sVelden(i)
Me("v" & i + 1).ControlSource = sVelden(i)
Me(12).ControlSource = sVelden(i)
```

Waarbij moet worden opgemerkt dat de laatste variant dus nauwelijks te gebruiken is, omdat je niet weet op welke plaats een bepaald object in de collectie staat.

The Icing on the Cake!

Nu alle elementen klaar zijn, kan het rapport getest worden. Als je het rapport hebt opgeslagen kunnen we de code voor de formulierknoppen afmaken. Daar ontbrak namelijk nog het commando om het formulier te sluiten en het rapport te openen. Die code luidt als volgt:

```
DoCmd.Close acForm, Me.Form.Name
```

```
DoCmd.OpenReport "rptKruistabel", acViewPreview

End Sub
```

Het formulier wordt gesloten, en met het commando OpenReport wordt het rapport < rptKruistabel> geopend. In het rapport zit nog een stukje code die het formulier <fKruistabel> weer opent met een schone lei. Wil je de selectie terugzien die je eerder had gemaakt, dan kun je het formulier ook verbergen, en bij het sluiten van het rapport weer zichtbaar maken.

Opdracht

Maak de ontbrekende code om het formulier te openen/zichtbaar te maken, en test het rapport!

Samenvatting

We hebben in dit hoofdstuk een zeer dynamisch rapport gemaakt, dat je kunt gebruiken om verschillende gegevensbronnen in een uniforme lay-out weer te geven. Op basis van een formulier waarin je de keuzes maakt wordt een tijdelijke query gevuld met een SQL-string, en daarna worden bij het openen van het rapport de veldnamen uit de query gekoppeld aan tekstvelden in het rapport.

Hoewel je met de hier behandelde techniek een heel flexibel rapport kunt bouwen, zit er wel een klein nadeel aan gekleefd: het maximale aantal velden dat je kunt koppelen ligt op voorhand wel vast, omdat je gebonden bent aan het aantal tekstvakken dat je vooraf hebt aangemaakt. Wil je daar ook nog flexibiliteit in, dan zul je het rapport met VBA in de Design modus moeten opmaken. Maar dat is een heel ander hoofdstuk…

De hier beschreven methodiek werkt in Access 2003- en Access 2007/2010-versie hetzelfde, en het is dus geen enkel probleem om een rapport dat op deze manier gemaakt wordt in een nieuwere of oudere versie te gebruiken.

Volgende Aflevering

In het volgende hoofdstuk gaan we ons specifiek bezighouden met de Rapporten in Access 2007/2010, omdat in die versies nogal wat is veranderd. Er is bijvoorbeeld een extra view bijgekomen, die een beetje schippert tussen een formulier en het oorspronkelijke rapport in, en waarin je ook gegevens kunt muteren. Zelf beschouw ik dat als een onzinnige optie die niet bij een Rapportage thuishoort; gegevens die je in een rapport ziet dienen bevroren te zijn. Als je een rapport voor een factuur ziet en die naar de klant verstuurt, dan mag je toch niet achteraf nog allerlei gegevens in die factuur wijzigen, lijkt mij! Maar het zit er nu in, dus laten we dan maar kijken of we er een nuttige toevoeging van kunnen maken…