Cursus Access - Hoofdstuk18

Handleiding van Helpmij.nl

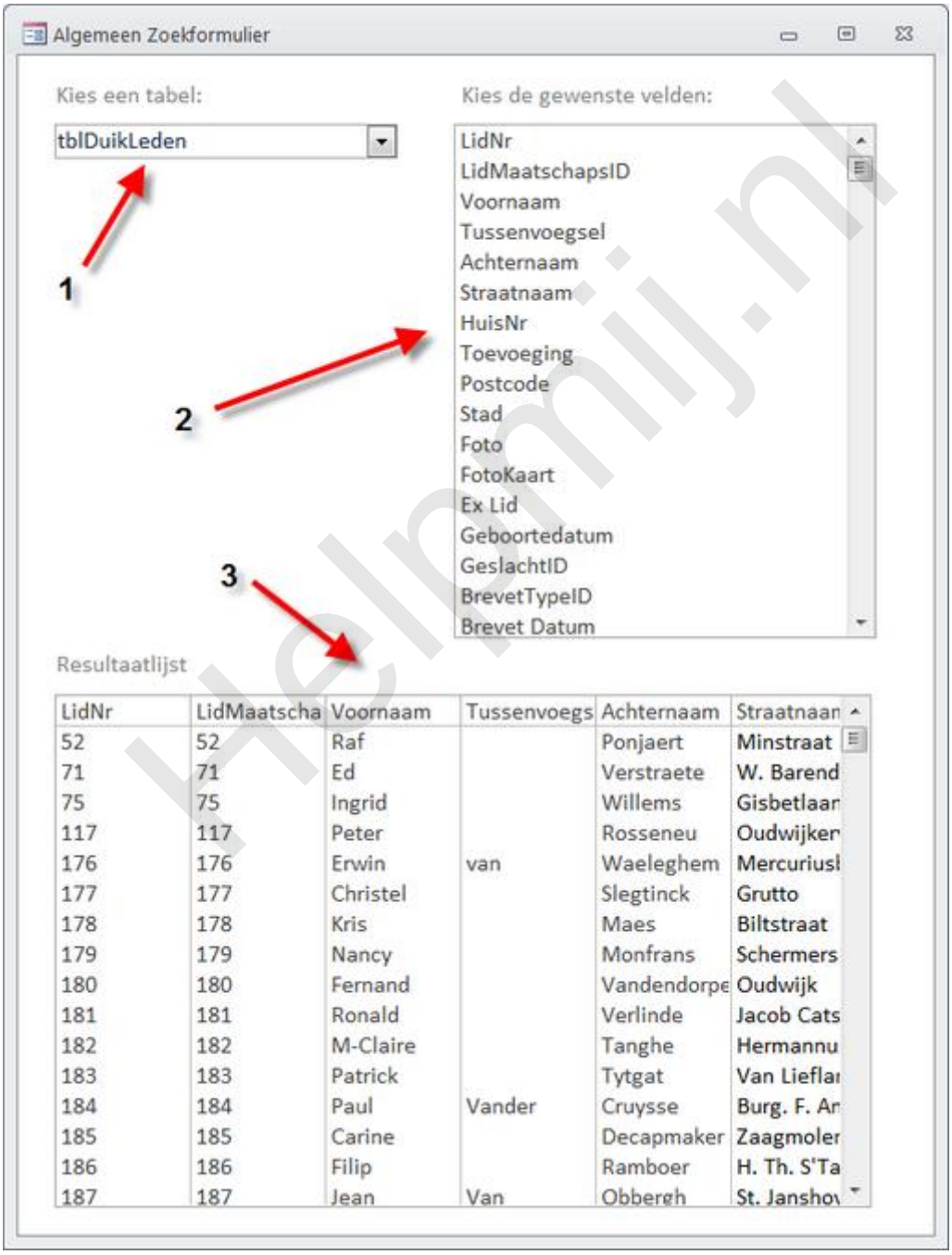
Auteur: OctaFish

Juli 2013

Een Zoekformulier

Heel veel Access gebruikers gebruiken zoekformulieren om gegevens op een formulier op te zoeken, en op basis daarvan een tweede formulier te openen met het gevonden record. En die constructie is natuurlijk prima. Maar als je veel tabellen en formulieren in je database hebt, zal je vermoedelijk ook meer zoekformulieren hebben, of meer zoekfuncties op bestaande formulieren.

In dit hoofdstuk gaan we één zoekformulier maken, dat je (bijna) overal voor kunt gebruiken. Het ziet er ongeveer zo uit:



Dit venster bestaat uit 3 objecten:

1. Keuzelijst met invoervak met keuze uit tabellen

2. Keuzelijst met beschikbare velden
3. Keuzelijst met resultaat van tabel met gekozen velden

De derde keuzelijst met het resultaat kun je dan weer gebruiken om bijvoorbeeld een formulier te openen. In eerste instantie is het zoekformulier bedoeld om records makkelijk op te zoeken, maar er zijn ook andere opties mogelijk, zoals een knop om de gekozen velden te exporteren, of een mailing te maken met de gekozen selectie.

1.1 Noodzakelijke voorwaarden

Wat heb je aan technieken nodig om dit formulier te maken? De basis keuzelijsten zijn relatief simpel. De eerste keuzelijst moet de beschikbare tabellen laten zien. Daarvoor zijn (minstens) twee methoden, die ik hier allebei behandel. Welke je neemt, maakt niet zoveel uit, al heb ik zelf een voorkeur voor de tabelmethode. Wel gebruiken we een andere manier voor het vullen van de keuzelijsten. Doorgaans gebruik je voor een keuzelijst de optie <Tabel/Query> als type, maar in dit formulier gaan we ook werken met <Lijst met velden>, en <Lijst met Waarden>.

De tweede voorwaarde is, dat je voordat je de derde keuzelijst kunt gebruiken, zeker moet weten dat hij ook correct gaat werken. En daarmee bedoel ik dit: als je de keuzelijst gebruikt om een record te openen op een nieuw formulier, dan wil je zeker weten dat het goede record wordt geopend. En dat betekent meestal, dat je dat formulier opent op basis van het *sleutelveld* van de tabel. En om dat goed te automatiseren, moet je dus altijd het sleutelveld terugzien in de resultaat keuzelijst (3). Alleen: hoe weet de gebruiker nu wat het sleutelveld (of de sleutelvelden) zijn in een tabel? Gelukkig hoeft hij dat niet te weten, want dat kunnen we met een functie achterhalen.

1.2 Keuzelijst om tabellen en/of queries te tonen

Zoals hierboven gezegd, zijn er twee methoden om te bepalen welke tabellen of queries we in een database hebben. De eerste methode maakt gebruik van collecties in de database, de tweede van een systeemtabel. We behandelen beide technieken, zodat je zelf kunt kiezen welke methode je voorkeur heeft. We gaan eerst keuzelijst (1) maken, die de tabellen laat zien.

1.2.1 Werken met collecties

Access heeft een eigenschap waarmee je kunt achterhalen welke objecten zoals tabellen, formulieren, rapporten en modules je in een database hebt opgeslagen. Door die collecties uit te lezen, en in een stringvariabele te zetten, kun je deze gegevens hergebruiken. Bijvoorbeeld als rijbron voor een keuzelijst.

De collecties waar we mee gaan werken zijn:

- AllTables
- AllQueries

Maar je hebt dus ook de collecties AllForms, AllReports, AllMacros en AllModules. Deze laatste zijn allemaal onderdeel van het object <**CurrentProject>**, terwijl wij dus gaan kijken naar objecten van het object *CurrentData*.

Laten we eens gaan kijken naar de functie.

```
Function TabelNamen() As String
Dim aObj As AccessObject

Dim dbs As CurrentData

Dim strVelden As String
```

```
    Set dbs = Application.CurrentData
    For Each aObj In dbs.AllTables
        If Left(aObj.Name, 4) <> "MSys" Then
            If strVelden <> "" Then strVelden = strVelden & ";"
            strVelden = strVelden & aObj.Name
        End If
    Next aObj
    TabelNamen = strVelden
End Function
```

We beginnen, zoals gewoonlijk, weer met het definiëren van de variabelen. Er zijn er drie. We kijken naar objecten, en daarom leggen we aObj vast als AccessObject. We willen in de collectie CurrentData kijken, dus daarom definiëren we dbs als CurrentData. En de laatste is een bekende...

Als eerste gebruiken we het SET commando om de variabele dbs te koppelen aan de eigenschap CurrentData. Die halen we uit de Application collectie.

Vervolgens lopen we met een lus door alle tabellen van het CurrentData object. We controleren eerst of de gevonden tabel geen (verborgen) systeemtabel is, want die tabellen hoeven we natuurlijk niet te zien in de keuzelijst. Het gaat ons om de tabellen met onze eigen gegevens.

Opmerking: systeemtabellen zijn altijd aanwezig in een database, maar meestal verborgen voor de gebruiker. Je kunt ze zichtbaar maken in het navigatiepaneel door de optie <Systeemobjecten weergeven> aan te zetten. Systeemobjecten beginnen altijd met de letters “MSys” dus door daar op te checken, kun je de systeemobjecten vermijden.

In de volgende IF...End If worden de juiste tabelnamen aan de string strVelden toegevoegd. En als laatste, als de lus is voltooid, wordt de variabele strVelden aan de functie toegewezen.

We hebben nu een string die we kunnen gebruiken om de keuzelijst te vullen.

Dat doen we bij het laden van het formulier, en wel op de volgende manier:

```
Private Sub Form_Load()
    With Me.cboTabel
        .RowSourceType = "Value list"
        .RowSource = TabelNamen
    End With
End Sub
```

We roepen bij het laden van het formulier dus de functie TabelNamen aan, die aan de RowSource van de keuzelijst wordt toegewezen.

1.2.2 Werken met systeemtabellen

We hebben het net al gezien in de procedure: Access kent systeemtabellen. Waar zijn die voor, en wat kunnen we er mee? De eerste vraag is simpel: Access slaat in de systeemtabellen allerlei variabelen op die het nodig heeft om de database te kunnen laten werken. In de nieuwere versies (vanaf 2007) worden de werkbalken die je zelf maakt gemaakt in XML, en de daarvoor gebruikte code vind je terug in de systeemtabel [MSysAccessXML]. De relaties die je legt in de database worden beschreven in de tabel [MSysRelationships]. En de tabellen, queries, formulieren etc. die je maakt worden opgeslagen in de tabel [MSysObjects]. En die laatste tabel kunnen we gebruiken om een lijst te genereren van de beschikbare tabellen.

Opmerking: Om de systeemtabellen te zien moet je ze zichtbaar maken door in het navigatiepaneel de optie <Systeemobjecten weergeven> aan te zetten.

Als je de systeemtabel [MSysObjects] bekijkt, zie je twee velden die voor ons interessant zijn: de velden]Name] en [Type]. De eerste bevat de *naam* van de objecten, de tweede het *soort object*. De tabel slaat namelijk alle mogelijke objecten op, en we zijn alleen geïnteresseerd in de tabellen. Als je door de systeemtabel bladert, dan zie je snel genoeg dat alle tabellen als type de waarde 1 hebben.

Wat we gaan doen is dus eigenlijk heel simpel: we gaan een query maken op basis van de tabel [MSysObjects] en we gaan die tabel filteren op het veld [Type]. De query ziet er daarom zo uit:

SELECT Name FROM MSysObjects WHERE ((Left([name],4)<>"MSys") AND ([Type]=1)) ORDER BY Name

Deze query wordt als Rijbron gebruikt voor de keuzelijst [cboTabel]. Die uiteraard als type <Tabel/query> heeft, want we gebruiken een query als rijbron.

Om deze query te kunnen gebruiken, hoef je de systeemobjecten overigens niet zichtbaar te hebben; je kunt deze query altijd maken en gebruiken. Al is het wel af en toe makkelijk om te kunnen zien wat je eigenlijk allemaal in de query aan het zetten bent. Maar als alles werkt zoals het moet, kun je de objecten weer verbergen.

Beide opties werken even goed; het is aan jou om te bepalen welke variant (of wellicht wil je ze beiden uitproberen) je kiest op je formulier.

1.3 De keuzelijst met Velden

Nu we een keuzelijst met invoervak hebben die de tabellen laat zien, kunnen we de aandacht verleggen naar keuzelijst (2), die de velden uit de gekozen tabel laat zien.

Deze keuzelijst is eigenlijk heel simpel. Je maakt daarvoor op je formulier een lege keuzelijst, die je groot genoeg maakt om een aantal velden te laten zien, en je laat hem leeg. Hij wordt dus niet gekoppeld aan een tabel. Dat doen we vanuit de eerste keuzelijst met invoervak. Wèl moeten we het <Type rijbron> aanpassen; dit wordt ingesteld op <Lijst met velden>. We willen immers de velden zien uit de gekozen tabel, en Access kan die automatisch ophalen.

We hebben dus een gebeurtenis nodig die we koppelen aan de actie <Bij klikken> van de keuzelijst <cboTabel>. En die ziet er zo uit:

```
Private Sub cboTabel_Click()
    With Me.lstVelden
        .RowSourceType = "Field list"
        .RowSource = Me.cboTabel.Value
        .Requery
    End With
End Sub
```

Het resultaat van dit alles is, dat de keuzelijst <Lijst met velden> de beschikbare velden laat zien die we kunnen gebruiken.

1.4 De resultaatlijst

De derde stap van het zoekformulier bestaat eruit dat we velden kiezen in de keuzelijst met velden, en op basis daarvan een keuzelijst vullen die de geselecteerde velden ophaalt uit de gekozen tabel. Hiervoor is het noodzakelijk om de eigenschap van de keuzelijst op Multiselect te zetten, anders kun je maar één veld selecteren. Tevens hebben we een functie nodig die controleert welk veld, of welke velden in de tabel de sleutel vormen, want die velden zijn in ieder geval nodig in keuzelijst (3). Want alles wat we willen doen met de lijst die we gaan maken, zal in ieder geval de sleutelvelden moeten hebben.

Verder zou het mooi zijn als de resultaatlijst dynamisch wordt aangepast aan de gekozen velden. In een tabel heb je doorgaans verschillende velden, met verschillende eigenschappen en verschillende veldlengtes. Dat laatste geldt dan voornamelijk voor tekstvelden, omdat numerieke velden in Bytes worden uitgedrukt. Maar ook daarvoor geldt natuurlijk dat je getallen en bedragen wel goed wilt kunnen zien in het resultaat.

1.4.1 De resultaatlijst vullen

Als eerste stap gaan we de code bekijken die we nodig hebben om de keuzelijst (3) te vullen. Hierbij heb je twee uitgangspunten. Je kunt een knop maken die één keer de resultaatlijst vult, of je vult de resultaatlijst elke keer bij als je een veld selecteert of deselecteert. Voor de techniek maakt dat niet zoveel uit, hooguit voor de snelheid van het formulier, omdat de tweede optie het bijwerken natuurlijk vaker uitvoert dan de knopversie. Hoe ziet de code om keuzelijst (2) uit te lezen er ook al weer uit (het onderwerp is al eens aan bod gekomen)? Laten we de code er eens bijhalen:

```
Private Sub lstVelden_Click()
Dim ctl As Control
Dim itm As Variant
Dim sVelden As String, strSQL As String
    For Each itm In Me.lstVelden.ItemsSelected
        If Me.lstVelden.ItemsSelected.Count > 0 Then
            If sVelden <> "" Then sVelden = sVelden & ", "
            sVelden = sVelden & "[" & Me.lstVelden.ItemData(itm) & "]"
        Else
            Exit Sub
        End If
    Next itm
    strSQL = "SELECT " & sVelden & " FROM [" & Me.cboTabel & "]"
    With Me.lstResultaat
        .RowSource = strSQL
        .RowSourceType = "Table/Query"
    End With
End Sub
```

Omdat deze techniek al eerder aan bod is geweest, hou ik de uitleg kort. Met For Each itm In Me.lstVelden.ItemsSelected starten we een lus die door alle geselecteerde waarden uit de keuzelijst loopt. We controleren eerst of er minstens één waarde is geselecteerd, want anders kunnen we geen query maken. Al zou je nog kunnen overwegen om de resultaatlijst te vullen met alle velden uit de tabel. Zie opdracht 1.

Bij voldoende (lees: minstens 1) geselecteerde waarde kunnen we een string gaan vullen met de geselecteerde waarden. Dat gebeurt met: sVelden = sVelden & "[" & Me.lstVelden.ItemData(itm) & "]"

Mochten er in de veldnaam spaties zitten, dan moeten er rechte haken om de veldnaam komen. Om het makkelijk te maken, zetten we die er standaard omheen. Hetzelfde zie je later ook gebeuren bij de tabelnaam. Als eerste in de IF...ELSE...END IF zie je nog een opdracht die de verschillende velden scheidt met een komma. Dit teken kan, afhankelijk van je landinstellingen, ook een puntkomma zijn. Controleer dus welk scheidingsteken je gebruikt als de code niet blijkt te werken.

Als alle geselecteerde items zijn uitgelezen, kunnen we een query maken die we als bron gebruiken voor de resultaatlijst. Als je al eens een query hebt gemaakt, zal de structuur je bekend voorkomen; het is een redelijk standaard Selectiequery. Eventueel kun je nog een sortering toevoegen op een veld. Probleem is natuurlijk dat je dan wel moet weten op wèlk veld je wilt sorteren. En dat maakt het nogal lastig.

Opdracht 1

Bouw de procedure lstVelden_Click zodanig om dat de keuzelijst lstResultaat alle velden uit de tabel laat zien als er geen waarde in de keuzelijst lstVelden is geselecteerd.

1.4.2 De sleutelvelden vinden

Een lastig klusje kan het opzoeken van de sleutelvelden zijn. Zoals al gezegd: wil je echt iets kunnen doen met het zoekresultaat, dan heb je vermoedelijk de sleutelvelden nodig om een ander formulier te openen.

Voor het opzoeken van de sleutelvelden maken we gebruik van de Index collectie. Eerst maar weer de functie, die we dan gaan ontleden.

```
Function PrimaryKey(tblName As String) As String
Dim db As DAO.Database
Dim td As DAO.TableDef
Dim idx As DAO.Index
Dim tmp As Variant
    Set db = CurrentDb
    Set td = db.TableDefs(tblName)
    For Each idx In td.Indexes
       If idx.Primary = True Then
        tmp = Replace(idx.Fields, "+", "")
          PrimaryKey = tmp
          Exit For
       End If
    Next idx
    db.Close
    Set db = Nothing

End Function
```

De functie PrimaryKey gebruikt een inputvariabel, namelijk de tabelnaam, en levert een string als resultaat. Dat betekent dat we het resultaat verder kunnen bewerken.

We vinden een nieuw variabele type: DAO.Index. De DAO bibliotheek is standaard altijd geladen in Microsoft, en hoeft als zodanig niet specifiek gedefinieerd te worden mits je alleen DAO gebruikt, en niet DAO en ADO tegelijk laadt. In het laatste geval (zoals in het voorbeeld) moet DAO specifiek in de verwijzing worden opgenomen.

Opmerking: DAO wordt door Microsoft al zo'n beetje vanaf de eerste versie van Access meegeleverd (versie 3.6) en is in al die jaren nauwelijks veranderd. Het DAO versienummer in Office 2010 is dus hetzelfde als in Access 97. Dat terwijl ADO steeds door evalueert, van ADO 2.1 naar (wederom Access 2010) ADO 6.1. Officieel ondersteunt Microsoft DAO ook al een aantal jaar niet meer, maar omdat veel gebruikers er nog steeds graag mee werken, en je er nog steeds zaken mee kunt regelen die in ADO niet kunnen (omgekeerd geldt overigens ook) blijft DAO dus gewoon in Access zitten.

Wat doet de functie precies? We wijzen eerst met SET de huidige database toe aan de variabele db, en vervolgens de tabeldefinitie van de tabel die als parameter is meegegeven aan de variabele td.

Daarna wordt met een lus door alle indexen gelopen die zijn opgeslagen in de tabel. Dat gebeurt met For Each idx In td.Indexes. Binnen de index collectie kan er maar één index primair zijn, en dat is ook degene die we zoeken, want dat is de primaire sleutel. De volgend regel is wellicht een beetje vreemd, maar logisch als je weet dat de naam van de index die je op deze manier uitleest altijd begint met een plust-teken. En daar hebben we niet zoveel aan, dus dat teken moeten we kwijt zien te raken. Vandaar dat er een tijdelijke variabele wordt gebruikt om de opgeschoonde index in op te slaan. Dat

gebeurt met de regel tmp = Replace(idx.Fields, "+", ""). Replace is in dit geval een snelle methode, omdat een index uit meerdere velden kan bestaan, en we elk veld natuurlijk willen opschonen.

Als we de primaire sleutel hebben gevonden, kunnen we deze toewijzen aan de functie, en de lus sluiten.

Hoe gaan we dit inbouwen in de veld procedure? Welnu, die moet een beetje worden aangepast.

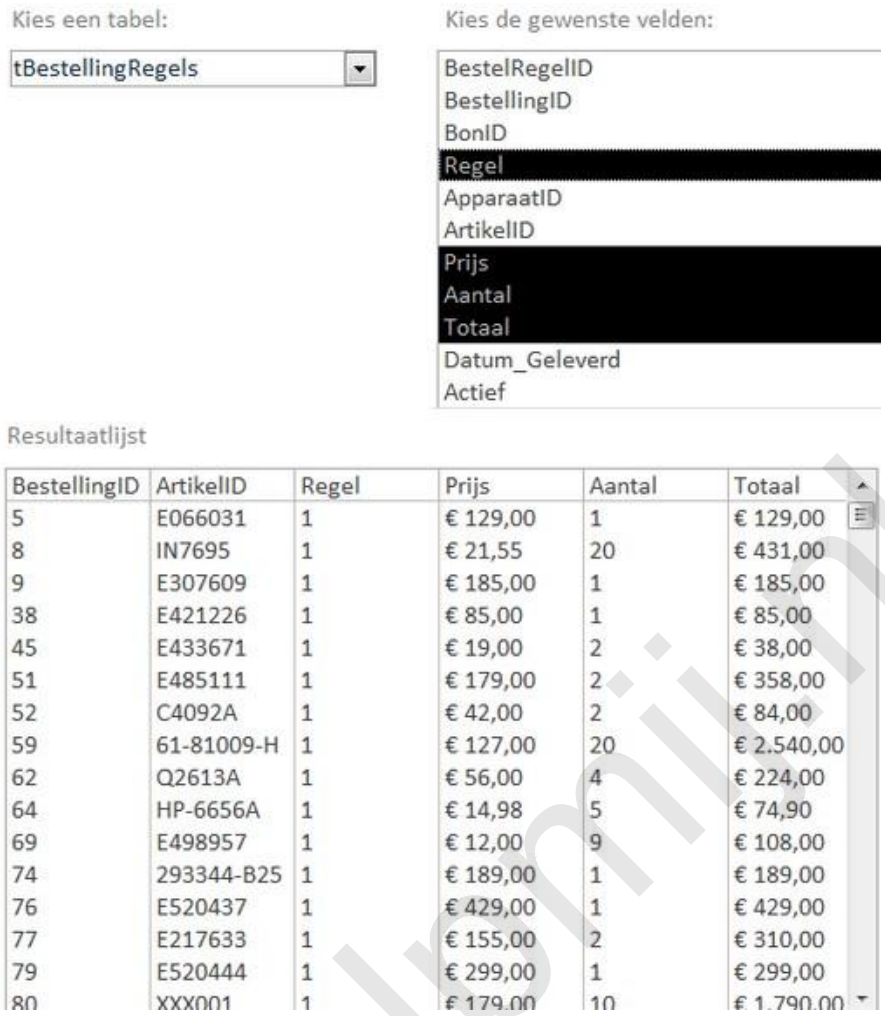
Om te beginnen, moet de nieuwe functie worden aangeroepen, en het resultaat in de variabele sVelden worden gezet. Vervolgens moet de keuzelijst worden uitgelezen zoals hierboven gebeurt, met als toevoeging dat we nu moeten controleren of de sleutelvelden niet ook geselecteerd worden. Want dan zouden ze dubbel in de lijst verschijnen, en dat is natuurlijk niet de bedoeling. We doen dat door de sleutelvelden in een matrix variabele te zetten (er kunnen tenslotte meerdere velden zijn) en die te vergelijken met de uitgelezen veldnaam. Als die in de matrix zit, is het een sleutelveld en slaan we 'm over. De aangepaste code ziet er nu zo uit, met de aanpassingen in vet:

```
sVelden = Replace(PrimaryKey(Me.cboTabel.Value), ";", ", ")
tmp = Split(sVelden, ",")
For Each itm In Me.lstVelden.ItemsSelected
    bCheckVeld = False
    If Me.lstVelden.ItemsSelected.Count > 0 Then
        For i = LBound(tmp) To UBound(tmp)
            If tmp(i) = Me.lstVelden.ItemData(itm) Then
                bCheckVeld = True
                Exit For
            End If
        Next i
        If bCheckVeld = False Then
            If sVelden <> "" Then sVelden = sVelden & ", "
            If InStr(1, Me.lstVelden.ItemData(itm), " ") > 0 Then
                sVelden = sVelden & "[" & Me.lstVelden.ItemData(itm) & "]"
            Else
                sVelden = sVelden & Me.lstVelden.ItemData(itm)
            End If
        End If
    Else
        Exit Sub
    End If
Next itm
```

De code begint nu met het vullen van de variabele sVelden met de sleutelvelden. Hier zie je een Replace in verwerkt, en dat is noodzakelijk (in mijn geval) omdat de velden in een index die uit meer velden bestaat gescheiden worden door een puntkomma. En de SQL string wordt opgebouwd met komma's, dus voor de zekerheid (je zult vaker een enkelvoudige sleutel vinden) worden de puntkomma's vervangen door komma's.

In de For Each lus wordt nu met SPLIT de variabele sVelden gesplitst, en wordt de geselecteerde waarde in een lus vergeleken met alle aparte sleutelveldnamen. Zo ja: dan heb je een sleutelveld aangeklikt, en wordt de variabele bCheckVeld op TRUE gezet. Zo nee (bCheckVeld = FALSE) dan zit het veld nog niet in de velden string, en voegen we 'm toe.

Het resultaat ziet er dan zo uit:



BestellingID	ArtikelID	Regel	Prijs	Aantal	Totaal
5	E066031	1	€ 129,00	1	€ 129,00
8	IN7695	1	€ 21,55	20	€ 431,00
9	E307609	1	€ 185,00	1	€ 185,00
38	E421226	1	€ 85,00	1	€ 85,00
45	E433671	1	€ 19,00	2	€ 38,00
51	E485111	1	€ 179,00	2	€ 358,00
52	C4092A	1	€ 42,00	2	€ 84,00
59	61-81009-H	1	€ 127,00	20	€ 2.540,00
62	Q2613A	1	€ 56,00	4	€ 224,00
64	HP-6656A	1	€ 14,98	5	€ 74,90
69	E498957	1	€ 12,00	9	€ 108,00
74	293344-B25	1	€ 189,00	1	€ 189,00
76	E520437	1	€ 429,00	1	€ 429,00
77	E217633	1	€ 155,00	2	€ 310,00
79	E520444	1	€ 299,00	1	€ 299,00
80	XXX001	1	€ 179,00	10	€ 1.790,00

Je ziet dat de eerste twee kolommen in het resultaat niet geselecteerd zijn. In deze tabel zijn dit de twee sleutelvelden.

1.4.3 De kolommen instellen

De laatste opdracht is absoluut niet nodig (we hebben immers een lijst met resultaten), en best lastig. Want we moeten niet alleen de veldeigenschappen uitlezen, we moeten ze ook nog eens vertalen naar zinvolle breedtes voor de kolommen. Kortom: het type klusje waar je bloed sneller van gaat stromen op een regenachtige woensdagmiddag!

We hebben hierbij te maken met een paar variabelen, zoals de breedte van de veldinhoud, de breedte van de kolomkoptekst en het gebruikte lettertype en lettergrootte. Daar komt nog bij dat kolommen in VBA in TWIPS worden gemeten, en dat is nu niet de meest handige maatsoort in het universum. Maar we kunnen een heel eind komen!

Dan hebben we ook nog de overweging *wanneer* de kolommen kunnen worden ingesteld. Logischerwijze kan dat pas als we weten welke velden we gaan gebruiken, en dat is dan inclusief de sleutelvelden die immers *altijd* voorkomen in de resultaatlijst. Want om in een keuzelijst kolombreedtes in te stellen, moet je de eigenschap ColumnWidths vullen met een string met de gewenste waarden. Kortom: we doen dat nà het instellen van de string met velden.

Als eerste moeten we een functie hebben waarmee we de maximale lengte van het veld kunnen opsporen. Dat kan uiteraard met een query, die we dan wel flexibel moeten maken. Vandaar ook dat we een functie gebruiken. Die zou er zo uit kunnen zien:

```
Function VeldGrootte(fldName As Variant, tblName As String) As Integer
Dim rs As DAO.Recordset
Dim strSQL As String
strSQL = "SELECT Max(Len(Nz(" & fldName & ")) AS n FROM [" & tblName & "]"
Set rs = CurrentDb.OpenRecordset(strSQL)
With rs
    If .RecordCount = 1 Then VeldGrootte = Nz(.Fields(0).Value, 0)
End With
End Function
```

Met deze functie kijken we hoeveel tekens er maximaal in een veld voorkomen. Dat gebeurt met een recordset, uiteraard weer gebaseerd op de gekozen tabel. De functie MAX levert hoe dan ook één record met één getal op, en die waarde zetten we in de functie VeldGrootte.

De veldgroottes die we met deze functie uitlezen zetten we in een matrix variabele, zoals je ondertussen van mij gewend bent, en we lopen daarna weer met een lus routine door alle velden heen. In elke gang van de lus roepen we de functie VeldGrootte aan, en het resultaat ervan zetten we in een tekststring. We vermenigvuldigen de waarde VeldGrootte met 150, om de veldlengte om te rekenen naar Twips. Maar die waarde is dus enigszins afhankelijk van het lettertype en de lettergrootte. Ook vergelijken we de veldlengte met de lengte van de veldnaam, om bij een te kleine waarde (wat bij getallen vaak het geval zal zijn) de veldlengte te berekenen op basis van de lengte van het label.

Als alle velden zijn uitgelezen, hebben we een string met getallen die de kolomwaarden bevatten.

```
With Me.lstResultaat
    .RowSource = strSQL
    .RowSourceType = "Table/Query"
    'Velden splitsen
    arr = Split(sVelden, ", ")
    For i = LBound(arr) To UBound(arr)
        tmp = arr(i)
        iBreedte = VeldGrootte(arr(i), Me.cboTabel)
        If iBreedte < Len(arr(i)) Then iBreedte = Len(arr(i))
        iBreedte = iBreedte * 150
        If sBreedte <> "" Then sBreedte = sBreedte & ";"
        sBreedte = sBreedte & iBreedte
    Next i
    .ColumnWidths = iBreedte
End With
```

De laatste truc om de resultaatlijst te verfraaien, is om de *breedte* van de keuzelijst aan te passen aan de totale waarde. En we moeten natuurlijk ook het *aantal zichtbare kolommen* in overeenstemming brengen met het aantal geselecteerde velden. De laatste twee aanpassingen zijn gelukkig simpel te maken. We kunnen namelijk een extra variabele maken die alle breedtes optelt. Die optelsom is automatisch de breedte die de keuzelijst moet hebben. Iets vergelijkbaars geldt voor het aantal zichtbare velden, want dat is afhankelijk van het aantal velden in de matrix. De totale code van de keuzelijst lstVelden ziet er derhalve zo uit:

```
Private Sub lstVelden_Click()
Dim ctl As Control
Dim arr As Variant, itm As Variant
Dim sVelden As String, sBreedte As String, strSQL As String
Dim iBreedte As Double, iTotaal As Double, i As Integer
Dim tmp As Variant
Dim bCheckVeld As Boolean
sVelden = Replace(PrimaryKey(Me.cboTabel.Value), ";", ", ")
tmp = Split(sVelden, ",")
For Each itm In Me.lstVelden.ItemsSelected
    bCheckVeld = False
```

```
    If Me.lstVelden.ItemsSelected.Count > 0 Then
        For i = LBound(tmp) To UBound(tmp)
            If tmp(i) = Me.lstVelden.ItemData(itm) Then
                bCheckVeld = True
                Exit For
            End If
        Next i
        If bCheckVeld = False Then
            If sVelden <> "" Then sVelden = sVelden & ", "
            If InStr(1, Me.lstVelden.ItemData(itm), " ") > 0 Then
                sVelden = sVelden & "[" & Me.lstVelden.ItemData(itm) & "]"
            Else
                sVelden = sVelden & Me.lstVelden.ItemData(itm)
            End If
        End If
    Else
        Exit Sub
    End If
Next itm
strSQL = "SELECT " & sVelden & " FROM [" & Me.cboTabel & "]"
With Me.lstResultaat
    .RowSource = strSQL
    .RowSourceType = "Table/Query"
    arr = Split(sVelden, ", ")
    iTotaal = 0
    For i = LBound(arr) To UBound(arr)
        tmp = arr(i)
        iBreedte = VeldGrootte(arr(i), Me.cboTabel)
        If iBreedte < Len(arr(i)) Then iBreedte = Len(arr(i))
        iBreedte = iBreedte * 150
        iTotaal = iTotaal + iBreedte
        If sBreedte <> "" Then sBreedte = sBreedte & ";"
        sBreedte = sBreedte & iBreedte
    Next i
    .ColumnWidths = sBreedte
    .Width = iTotaal
    .ColumnCount = UBound(arr) + 1
End With
End Sub
```

Opdracht 2:

De kolombreedtes zijn nog niet afhankelijk van het lettertype. Vervang de berekening voor iBreedte door een berekening die rekening houdt met de lettergrootte.

Opdracht 3:

Breid het formulier uit met een optie om of een tabel, of een query of allebei te kunnen kiezen als recordbron. Kijk hiervoor in de tabel MSysObjects om te achterhalen welke waarde een query heeft.

Samenvatting:

We hebben in dit hoofdstuk een formulier gebouwd dat uiterst flexibel resultaten kan laten zien uit een tabel. We gebruiken daarvoor een aantal verschillende technieken om keuzelijsten te vullen. De Resultaatlijst wordt ‘live’ gevuld met de gekozen velden. Dit formulier kun je als basis gebruiken voor vervolgacties, zoals het openen van een formulier met een geselecteerd record, of het sturen van een emailbericht naar een aantal geselecteerde personen uit de resultaatlijst.