Access Voor Beginners - Hoofdstuk 8

Handleiding van Helpmij.nl

Auteur: OctaFish

September 2011

Formulieren: (Afhankelijke) keuzelijsten maken

In het vorige hoofdstuk hebben we een begin gemaakt met het doorgronden van queries; in hoofdstuk 5 hebben we een begin gemaakt met formulieren en in hoofdstuk 6 hebben we kennisgemaakt met programmeren. In dit hoofdstuk gaan we deze hoofdstukken combineren, door afhankelijke keuzelijsten (met invoervak) te maken.

In dit hoofdstuk gaan we zoals gezegd een formulier maken met keuzelijsten, die zijn gebaseerd op queries. Dat klinkt al gelijk moeilijker dan het is, want bijna elke keuzelijst die we maken is gebaseerd op een query, al kun je voor keuzelijsten (al dan niet met invoervak) ook een tabel als basis gebruiken. Maar een query als bron voor een keuzelijst biedt veel meer mogelijkheden dan alleen een enkele tabel.

Een keuzelijst om records op te zoeken

Een van de eerste keuzelijsten die een beginnende Access-gebruiker meestal maakt, is de keuzelijst om records te zoeken op een formulier. Deze keuzelijst is relatief simpel te maken met de wizard; de wizard doorloopt een aantal stappen en aan het eind ervan heb je een keuzelijst gemaakt. Deze keuzelijst gaan we dan ook als eerste maken en vervolgens analyseren.

Een keuzelijst waarmee je records opzoekt kan op allerlei formulieren worden gebruikt, denk aan het formulier voor de ledenlijst bijvoorbeeld. Als je een bepaalde persoon zoekt, kun je uiteraard de zoekfunctie gebruiken, maar zoeken in een formulier is min of meer hetzelfde als zoeken in een tabel: je moet eerst het juiste veld selecteren en bij het zoeken exact opgeven hoe er moet worden gezocht. Weet je bijvoorbeeld alleen de beginletters van de naam zeker, dan wil je niet zoeken op de volledige inhoud van het veld, omdat je de persoon dan niet vindt. Keuzelijsten geven de overzichten op basis van de velden die we selecteren en in het gebruik is het meestal al genoeg om de eerste letters te typen om het juiste record te vinden.

Laten we eens een keuzelijst maken op het formulier <f_Leden> waarmee we een persoon kunnen opzoeken. Let er op dat de knop Wizard () actief is.

1. Klik op de knop **Keuzelijst met invoervak** ()
   De muisaanwijzer verandert in een mini-keuzezlijst.
2. Klik in het lege rechterdeel van de koptekst
   De wizard *Keuzelijst met Invoervak* verschijnt.

3. Kies de optie **Een record in het formulier opzoeken op basis van de waarde die is geselecteerd in de keuzelijst met invoervak** en klik op **Volgende**

In de tweede stap van de wizard geef je aan welke velden je wilt zien in de keuzelijst. Dat kan één veld zijn, maar je kunt ook meer velden selecteren, zodat je het op te zoeken record makkelijker kunt herkennen.

4. Kies het veld Naam, en klik op de knop Toevoegen ()

Het veld *Naam* wordt verplaatst van de kolom *Beschikbare velden* naar de kolom *Geselecteerde velden*.

5. Voeg ook de velden *Voornaam, Straatnaam* en *Plaats* toe aan de kolom *Geselecteerde velden* en klik op **Volgende**

Je ziet nu de kolommen die je straks in je keuzelijst terug ziet.

Let op het vinkje boven de kolommen! Access voegt voor deze keuzelijst automatisch het sleutelveld toe, en stelt de eigenschap daarvan in op *Verborgen*. Je kunt het vinkje uitzetten om te zien welk veld je straks niet ziet, maar het is inderdaad aan te bevelen om het veld te verbergen. Zorg er dus voor dat het vinkje aan staat voordat je op *Volgende* klikt.

In het scherm kun je, net zoals je dat in Excel en in je tabellenvenster gewend bent, op het oog de kolombreedtes aanpassen. Neem daar even de tijd voor, want dat scheelt straks een hoop tijd als je achteraf alsnog andere kolombreedtes zou willen. Je moet dan namelijk de breedte opgeven in cm's, en niet meer op het oog. Dit venster geeft je dus de mogelijkheid om visueel de kolommen in te delen!

1. Als je tevreden bent over de lay-out, klik je op de knop **Volgende**
2. Typ in het laatste scherm de tekst die je bij de keuzelijst wilt zien, en klik op **Voltooien**. Je keuzelijst is nu klaar voor gebruik.
3. Open het formulier in de *Formulierweergave*, en klik op de knop rechts in de lijst. Selecteer een record. Je ziet het geselecteerde record op het formulier verschijnen.

De keuzelijst is nu gemaakt, maar wat heeft Access eigenlijk gedaan? Tijd om de keuzelijst eens aan een nader onderzoek te onderwerpen!

Eigenschappen van een keuzelijst: de Rijbron

Om de eigenschappen van de keuzelijst te bekijken, moeten we uiteraard eerst terug naar het Ontwerpscherm; alleen daarin kunnen we de eigenschappen goed bekijken en aanpassen.

Klik op de keuzelijst met invoervak om die te selecteren, en klik vervolgens op de knop <Eigenschappen>. Klik vervolgens op het tabblad <Gegevens>. Je ziet het volgende scherm:

Bovenin zie je een keuzelijst met daarin alle objecten die je kunt selecteren, je kunt namelijk van bijna alle objecten wel iets instellen of bekijken. Door ergens in het formulier iets te klikken verander je dus ook het eigenschappenvenster, want je ziet dan de eigenschappen van het geselecteerde object. Een makkelijke manier dus om eigenschappen van welk object dan ook te bekijken!

We kijken, zoals gezegd, nu dus naar de eigenschappen van de nieuwe Keuzelijst met Invoervak die we hebben gemaakt. Die heeft, zoals je kunt zien, een nogal belabberde naam gekregen van Access. Jammer genoeg heeft het aanpassen van de naam vervelende consequenties, maar omdat we het formulier graag overzichtelijk en onderhoudbaar willen hebben, gaan we dat straks toch wel doen. Maar daarover later meer…

Het type rijbron

Waar het in deze paragraaf om gaat, is dus de regel *Rijbron*. In lichte samenhang daarmee is de regel erboven: *Type rijbron*. Deze bepaalt namelijk wat voor soort keuzelijst we hebben gemaakt. Er zijn drie types: *Tabel/Query*, *Lijst met waarden* en *Lijst met velden*.

- *Tabel/Query*
  Deze optie gebruik je als je de keuzelijst op een tabel of een query wilt baseren. Dit is waarschijnlijk de meest voorkomende toepassing die je zult gebruiken.
- *Lijst met waarden*
  Als je maar een paar vaste waarden gebruikt (denk aan een keuzelijst voor Geslacht, met maar een paar opties) kun je deze optie overwegen. Hierbij typ je in de regel *Rijbron* zelf welke opties je wilt zien. Denk nog even terug aan de wizard, waarin in stap 1 gevraagd werd hoe je de lijst wilde maken, optie 2 daarin was: de waarden zelf typen.
- *Lijst met velden*
  Deze optie gebruik je als je in een formulier velden moet kunnen kiezen. Denk daarbij bijvoorbeeld aan een formulier waarbij de gebruiker zelf een query mag samenstellen; met deze optie kan hij/zij dan de velden selecteren voor de query.

De rijbron nader bekeken

In het voorbeeld hebben we een query gemaakt op basis van een tabel. Deze query zien we terug in de regel *Rijbron*. Als we deze regel selecteren, en op <Shift>+<F2> drukken, zien we de regel in zijn volle glorie:

SELECT LidNr, Naam, Voornaam, Straatnaam, Stad FROM st_Leden

ORDER BY Naam, Voornaam;

Ik heb de code een klein beetje opgeschoond; Access zet altijd de tabelnaam voor de veldnaam, maar meestal is dat niet nodig. Als je zelf een SQL code typt, kun je dus doorgaans volstaan met het intypen van de veldnaam zoals hierboven.

Je ziet dat er 5 velden zijn geselecteerd, al hebben we er zelf maar 4 toegevoegd. Het veld [Lidnr] is door Access toegevoegd, en dat is dus het (verborgen) sleutelveld dat je in de wizard bent tegengekomen. Dit veld is nodig om de keuzelijst goed te laten werken, maar dat zien we straks wel. Je ziet in de code ook dat deze keuzelijst wordt gesorteerd op twee velden: het veld [Naam], en vervolgens op het veld [Voornaam]. Je weet waarschijnlijk nog wel dat Access de records eerst sorteert op het eerste veld, daarna op het volgende en zo verder.

Je kunt de SQL ook bekijken in de query ontwerp weergave; daarvoor klik je op de knop met de drie puntjes ( ). In het query ontwerp venster kun je op dezelfde manier aan de query werken als eerder in het hoofdstuk Queries is behandeld.

Tip:
In de wizard (en dat geldt ook voor een aantal andere wizards) is het niet mogelijk om meer dan één tabel te gebruiken. In dat geval maak je de query op basis van de hoofdtabel en voeg je de overige velden uit de andere tabellen of queries later toe, door de rijbron van de keuzelijst aan te passen.

De standaardwaarde van de keuzelijst

Elke keuzelijst heeft een veld nodig dat als waarde wordt geretourneerd als je de Waarde (Value) van de keuzelijst opvraagt. Deze waarde wordt bepaald door de eigenschap <*Afhankelijke kolom*>. In ons voorbeeld staat die waarde op 1, zijnde het sleutelveld [Lidnr]. Als je gaat schuiven met de volgorde van de velden, dan zul je waarschijnlijk ook de waarde in *Afhankelijke kolom* moeten aanpassen, omdat de keuzelijst anders wel eens vreemd kon gaan werken…

De opmaak van de keuzelijst aanpassen

Keuzelijsten die je met een wizard maakt, zien er niet altijd even 'gelikt' uit; vaak kun je de opmaak van de keuzelijst nog wel wat verbeteren. Dat doe je op het tabblad *Opmaak*.

Als we bovenaan beginnen, dan zien we de volgende interessante opties:

- *Aantal kolommen*
  In de keuzelijst staat nu het aantal 5; dat komt uiteraard overeen met het aantal velden in de query.
- *Kolombreedten*
  Hier staan de feitelijke afmetingen van de keuzelijst. In het voorbeeld:
  0cm;1,985cm;1,614cm;2,963cm;1,932cm.
  Je ziet dat de lijst begint met 0cm: hiermee wordt de breedte van de eerste kolom ingesteld op 0, en dat maakt die kolom dus in de praktijk onzichtbaar. Je kunt net zoveel kolommen verbergen als je wilt; je hoeft alleen de kolombreedte dus op 0cm in te stellen.
  Je ziet ook de specifieke waarden zoals je die hebt ingesteld bij het slepen van de kolommen.
- *Aantal rijen*
  Dit is een interessante instelling; bij Access 2003 wordt hier standaard een waarde van 8 ingesteld; dit is meestal een erg lelijke waarde. Access 2007/2010 gebruikt standaard de waarde 16 en dat ziet er op het scherm al veel beter uit. Zelf gebruik ik vaak de waarde 30, omdat je op het scherm vaak wel zoveel rijen tegelijk kunt laten zien zonder het scherm te verlaten met de keuzelijst. Instellen naar eigen inzicht dus, deze optie!
- *Lijstbreedte*

Deze waarde is wel interessant om goed in te stellen; bij een te kleine waarde zal Access een horizontale scrollbalk nodig hebben om alle kolommen te laten zien. Een goede vuistregel, de keuzelijst die exact de goede breedte geeft is deze:

Alle kolombreedtes opgeteld + 0,6 cm.

Als ik dat toepas op de volgende kolombreedtes: *0cm;2cm;1,5cm;3cm;2cm* dan is de ideale lijstbreedte voor deze keuzelijst: 0+2+1,5+3+2+0,6 = **9,1cm**.

- *Hoogte*
  Hiermee stel je de hoogte in van de keuzelijst. Je kunt de waarde hier veranderen, maar je kunt de hoogte ook gelijk maken aan andere objecten in het formulier via het menu <Opmaak>, <Grootte>, <Aan langste> bijvoorbeeld.

De overige opties spreken redelijk voor zichzelf, en daar mag je dus op eigen gelegenheid mee gaan stoeien…

De gebeurtenis <Na bijwerken> van de keuzelijst

De keuzelijst doet wat, hebben we geconstateerd: hij zoekt een specifiek record op. En wel het record dat we hebben geselecteerd. Hoe doet hij dat? Daarvoor moeten we kijken naar het tabblad *Gebeurtenis*.

Elk object kan op verschillende momenten een actie uitvoeren; welke dat zijn, hangt af van het soort object. Zo kun je bij een label bijvoorbeeld slechts de gebeurtenissen <*Bij klikken*>, <*Bij dubbelklikken*>, <*Bij muis omlaag*>, <*Bij muis verplaatsen*> en <*Bij muis omhoog*> gebruiken. Bij een tekstvak zijn deze opties ook beschikbaar, maar daar heb je bijvoorbeeld ook de gebeurtenissen <*Voor bijwerken*> en <*Na bijwerken*>. En voor knoppen gelden weer andere gebeurtenissen. Het is dus belangrijk, als je een gebeurtenis wilt maken, dat je het juiste object selecteert voordat je begint!

Bij onze keuzelijst vinden we één gebeurtenis aangeduid: namelijk de gebeurtenis <*Na bijwerken*>. En dat kan ook wel, want er gebeurt iets *nadat* we een keuze hebben gemaakt in de keuzelijst. De code van de keuzelijst kunnen we bekijken door (daar is-tie weer) de knop met de puntjes ( ) aan te klikken. Daarvoor moet je eerst in de regel <*Na bijwerken*> klikken. Je komt nu in het VBA venster terecht, waar deze code staat:

```
Private Sub Keuzelijst_met_invoervak58_AfterUpdate()

' De record zoeken die overeenkomt met het besturingselement

Dim rs As Object

Set rs = Me.Recordset.Clone
rs.FindFirst "[LidNr] = " & Str(Nz(Me![Keuzelijst met invoervak58], 0))

   If Not rs.EOF Then Me.Bookmark = rs.Bookmark

End Sub
```

De eerste naam bevat de naam van de keuzelijst (Keuzelijst_met_invoervak58) gevolgd door de functie die de procedure uitvoert (_AfterUpdate()). Daaronder een regel commentaar, die aangeeft wat de keuzelijst doet, en daaronder een regel die een object vastlegt: het object *RS*. Deze variabele wordt verderop gevuld.

Met de regel Set rs = Me.Recordset.Clone wordt de tabel of query die op het formulier gebruikt wordt toegewezen aan de variabele *rs*. Hierbij wordt een *kopie* van de tabel of query gebruikt. In de zoekroutine wordt straks elk record uit de kopietabel vergeleken met het record dat gezocht moet worden. Er zijn dus twee tabellen (recordsets) nodig om de functie te kunnen laten werken: de records van het formulier en de records in de zoektabel.

In het hoofdstuk over VBA is het gebruik van het woord ME al aangestipt; hiermee wordt het actieve object uitgelezen. Dat kan een tekstvak zijn (Me.txtAchternaam bijvoorbeeld) of het

bovenliggende niveau, in dit geval dus het formulier zelf. De regel Set rs = Me.RecordsetClone doet dus exact hetzelfde, en is zeker in het begin misschien wat duidelijker in zijn werking.

De volgende regel bepaalt wat er gaat gebeuren: er wordt namelijk een record opgezocht op basis van de waarde die in de keuzelijst is geselecteerd. Dat opzoeken gebeurt met het commando rs.FindFirst. Wat er vervolgens wordt gezocht staat hier: "[LidNr] = " & Str(Nz(Me![Keuzelijst met invoervak58], 0)). Oftewel: zoek het lidnummer op (veld [Lidnr]) op basis van de waarde uit de keuzelijst. De gekozen waarde wordt door Access nog een beetje ‘bewerkt’; dit is meestal overbodig, en de code kan er dus ook zo uitzien:
"[LidNr] = " & Me.[Keuzelijst met invoervak58]. Om de syntax helemaal 100% te krijgen, zou je ook deze variant kunnen gebruiken: "[LidNr] = " & Me.[Keuzelijst met invoervak58].Value. De toevoeging *Value* is echter niet noodzakelijk; elk object heeft wel zijn eigen standaardinstelling. Als je een object aanroept in een routine, en verder niets specificeert over dat object, dan gaat Access er vanuit dat je de standaardeigenschap bedoelt. Bij een keuzelijst is dat dus de eigenschap *Value*.

De laatste regel voert het commando uit. Me.Bookmark = rs.Bookmark. **Er zit nog wel een beperking bij: alleen naar het record zoeken als niet het record niet het laatste record is gevonden.** Zonder deze restrictie zou Access door blijven zoeken, en dat wil je uiteraard niet. Je ziet hier ook waarom we met een kopie van de tabel moeten werken: je kunt anders het record op het formulier niet vergelijken met het gevonden record in de zoektabel.

De keuzelijst: de laatste stap.

Samenvattend kunnen we dus het volgende zeggen over de keuzelijst: we hebben gezien hoe we de opmaak van de keuzelijst kunnen aanpassen, zodat hij er nog beter uit komt te zien, wat het gebruik ervan alleen maak kan stimuleren. We hebben gezien hoe we de gegevens die in de keuzelijst staan kunnen beïnvloeden, door de *Rijbron* aan te passen. En via het tabblad *Gebeurtenis* hebben we gezien welke actie de keuzelijst uitvoert. De keuzelijst heeft alleen nog geen mooie naam, en eigenlijk wil je dat ook wel geregeld hebben; straks heb je misschien veel meer keuzelijsten op je formulier, en met namen als [Keuzelijst met invoervak58] wordt je over de functie van de keuzelijst ook niet veel wijzer. Want welke keuzelijst bedoelen we hiermee? Ik vind het derhalve uiterst nuttig om de naam van de keuzelijst aan te passen. Dat doe ik als volgt:

In de regel Private Sub Keuzelijst_met_invoervak58_AfterUpdate() verander ik de tekst vóór het _ teken naar de naam die ik wil gebruiken. In dit geval wil ik als naam: **cboZoekLijst**. De regel wordt dus: Private Sub cboZoekLijst_AfterUpdate().

Omdat de functie de geselecteerde waarde uit de zoeklijst opzoekt, moet ik dat in de routine ook aanpassen. De regel "[LidNr] = " & Me.[Keuzelijst met invoervak58]wordt dus:
"[LidNr] = " & Me.[cboKeuzeLijst]. De complete routine ziet er uiteindelijk zo uit:

```
Private Sub cboZoekLijst_AfterUpdate()
  ' De record zoeken die overeenkomt met het besturingselement
   Dim rs As Object
   Set rs = Me.Recordset.Clone
   rs.FindFirst "[LidNr] = " & Me.cboZoekLijst.Value
   If Not rs.EOF Then Me.Bookmark = rs.Bookmark
End Sub
```

Er is nog één handeling nodig: het daadwerkelijk aanpassen van de *Naam* van de keuzelijst. En dat doe je op het tabblad <*Overige*> de naam van de keuzelijst te veranderen in de regel *Naam*. In het voorbeeld wordt dat dus: **cboZoekLijst**.

Als je het formulier nu opslaat, zul je zien dat alle wijzigingen netjes worden opgeslagen, en doet de

keuzelijst nog steeds waarvoor je hem gemaakt hebt.

Keuzelijst om records te filteren

De keuzelijst die we in de vorige paragraaf hebben gemaakt fungeert prima op een formulier in *Enkelvoudige weergave*, met één record op het formulier. De keuzelijst zoekt één record op, en gebruikt daarvoor het sleutelveld van de tabel waarin we zoeken. Dat sleutelveld is, hebben we gezien, ongevraagd door Access toegevoegd aan de velden die we zelf hebben geselecteerd om in te zoeken. En, dat hebben we al veel eerder gezien, een sleutelveld heeft als eigenschap dat het een unieke waarde is. Ergo: deze waarde is in de tabel maar één keer te vinden. Wat dat betreft hadden we in de code net zo goed rs.FindLast kunnen gebruiken, want of je nu de eerste waarde zoekt of de laatste: als er maar één van is te vinden, vind je geen ander.

Anders wordt het als je records wilt zoeken op een *Doorlopend formulier*; weliswaar werkt de keuzelijst daar op exact dezelfde manier, maar het overzicht op het formulier is niet geweldig; je ziet namelijk nog steeds alle records, alleen staat het gezochte record nu, als het al niet op het formulier zichtbaar was, bovenaan in de lijst. Op een doorlopend formulier is een *Filter* veel nuttiger. Met een filter laat je alle records zien die overeenkomen met de waarde die je in (bijvoorbeeld) een keuzelijst selecteert.

Een formulier heeft een eigenschap waarmee je een filter kunt instellen: de eigenschap *Filter*. Dat filter kunnen we uiteraard instellen m.b.v. VBA, en dat gaan we dus doen in de volgende oefening. Als basis gebruiken we een formulier dat we hebben gebaseerd op de Duiklocaties. Op dit formulier zien we een lijst van alle duiklocaties, netjes geschikt op land en locatie. Een deel van dat formulier ziet er zo uit:

Aruba	Caribische Zee	Spanish Lagoon Reef
Aruba	Caribische Zee	Sponge Reef
Aruba	Caribische Zee	The Cross
Aruba	Caribische Zee	The Finger
Aruba	Caribische Zee	Tire Reef
Aruba	Caribische Zee	Antilla Wreck
Aruba	Caribische Zee	Lago Reef
Aruba	Caribische Zee	Six Sisters
Egypte	Dahab	Red Tooth Trigger Bay
Egypte	Dahab	Bannerfish Bay
Egypte	Dahab	Blue Hole & The Bells
Egypte	Dahab	Canyon & Fish Bowl
Egypte	Dahab	Caves
Egypte	Dahab	Eel Garden

De totale lijst bevat in dit voorbeeld 296 records, dus als je een bepaalde locatie zoekt, is het bladeren door alle records behoorlijk omslachtig. Beter is het dan om de lijst te filteren op Land, of Locatie. Of allebei.

Unieke waarden laten zien in een keuzelijst

Als we beginnen met de keuzelijst uit de vorige oefening, en de keuzelijst maken om een Land op te zoeken, dan krijgen we in eerste instantie een zoeklijst die alle 296 landen laat zien, wel netjes gesorteerd op land, want dat hebben we uiteraard ingesteld. Maar om eerst door tientallen landen te bladeren om Nederland te vinden, is uiteraard niet handig. De eerste stap is dus om de SQL van de keuzelijst aan te passen. Dat kan heel simpel door een kleine aanpassing te maken in de code van de query. Die luidt nu:

```
SELECT  LandNaam FROM qDuikplaatsen

ORDER BY LandNaam;
```

Zoals je ziet, een eenvoudige Select query, die het veld [Landnaam] uit de bron pakt, en die gesorteerd laat zien. Door aan deze query het woord DISTINCT toe te voegen, veranderen we een belangrijke eigenschap van de query: hij laat dan namelijk niet meer *alle waarden* zien, maar alleen de *unieke waarden*. En dan is de lijst gelijk een stuk korter, want dan komen Aruba en Egypte nog maar één keer voor in de lijst. De code ziet er dan zo uit:

```
SELECT  DISTINCT LandNaam FROM qDuikplaatsen

ORDER BY LandNaam;
```

Je hoeft, om deze aanpassing te maken, de query dus niet eens te openen, het typen van het woord DISTINCT achter het woord SELECT is al genoeg om de keuzelijst aan te passen. De keuzelijst ziet er nu zo uit:

En dat is uiteraard een stuk makkelijker zoeken! Maar de lijst is nog net zo onhandig in het gebruik als voorheen, want de actie die wordt uitgevoerd is nog steeds: zoek het eerste record dat overeenkomt met de geselecteerde waarde. Die code is:

```
Private Sub cboLand_AfterUpdate()
    ' De record zoeken die overeenkomt met het besturingselement
    Dim rs As Object
    Set rs = Me.Recordset.Clone
    rs.FindFirst "[LandNaam] = '" & Me.cboLand & "'"
    If Not rs.EOF Then Me.Bookmark = rs.Bookmark
End Sub
```

We willen, zoals gezegd, echter een selectie maken op het formulier, en niet een record opzoeken. We maken daarom een paar kleine aanpassingen aan de code. Om te beginnen, hebben we het Recordset object niet meer nodig. Die code kan dus weg.

In plaats van de opdracht rs.FindFirst maken van de *Zoekstring* een *Filterstring*, door rs.FindFirst te vervangen door Me.Filter = De code ziet er dan zo uit:

```
Private Sub cboLand_AfterUpdate()
    Me.Filter = "[LandNaam] = '" & Me.cboLand & "'"
End Sub
```

Je ziet, dat de code voor het filteren in wezen hetzelfde is als de code voor het zoeken! Daarom gebruik ik als basis voor een bepaalde keuzelijst nog wel eens een keuzelijst die met een wizard is gemaakt; vaak is de basiscode namelijk heel makkelijk om te zetten naar het eigenlijke doel waarvoor de keuzelijst is bedoeld.

Overigens is de code nog niet af; het filter is nu weliswaar aan het formulier gekoppeld, maar het filter is nog niet actief. Daarvoor is een extra regel nodig: Me.FilterOn=True. De totale code ziet er dus zo uit:

```
Private Sub cboLand_AfterUpdate()
    Me.Filter = "[LandNaam] = '" & Me.cboLand & "'"
    Me.FilterOn = True
End Sub
```

Als je de keuzelijst nu uitprobeert, zul je zien dat de lijst netjes wordt gefilterd met het land dat je hebt gekozen. In de navigatorbalk van het formulier zie je dat ook terug: daar staat niet meer het totale aantal records, maar het aantal gefilterde records.

Opdracht

Maak op dezelfde manier een keuzelijst waarmee je het formulier filter op *Locaties*.

Keuzelijsten afhankelijk van elkaar maken

De keuzelijst Locatie werkt op dezelfde manier als de keuzelijst voor Landen, maar heeft toch een nadeel: als je een land kiest, kun je nog steeds locaties kiezen uit andere landen. En als je een locatie selecteert, dan verandert dus ook het land, want een locatie is uiteraard maar aan één land gekoppeld. Het zou logischer zijn, als de lijst met Locaties alleen díe locaties zou laten zien die in het geselecteerde land liggen, zodat je niet een combinatie kunt maken die onmogelijk is. Oftwel: als ik *Nederland* kies, dan wil ik alleen de opties *Grevelingen* en *Oosterschelde* kunnen kiezen. De tweede keuzelijst moet dus *afhankelijk* worden gemaakt van de eerste keuzelijst.

Daarvoor zijn twee technieken beschikbaar, die ik alletwee zal behandelen. Welke van de twee je zelf gaat toepassen, mag je straks zelf uitmaken…

Methode 1: een criterium maken in de SQL van de keuzelijst

Bij deze methode passen we de query-eigenschappen van de keuzelijst aan. Deze methode is relatief simpel uit te voeren, en levert weinig VBA-code op, zoals je zult zien. Het principe van deze methode is, dat we de waarde van de eerste keuzelijst (Landen) als criterium gebruiken in de SQL van de tweede keuzelijst. We moeten dus de Rijbron van de keuzelijst aanpassen.

- Selecteer de Eigenschappen van de keuzelijst <*cboLocatie*>, klik op het tabblad <**Gegevens**>, klik in de regel <**Rijbron**> en klik op de knop met de puntjes ( )
- In het query-ontwerpscherm: voeg het veld [*Landnaam*] toe aan het raster
- Klik in de regel Criteria, en klik op de knop Opbouwen ( ) (of druk op <Ctrl>+<F2>)
- In de functie *Opbouwen voor Expressies* dubbelklik je op **Forms**, **Geladen formulieren** en selecteer het formulier waarin je de keuzelijst maakt.
- Zoek in de tweede kolom in de lijst met <Formulier> objecten de keuzelijst *cboLanden* op, en klik op **Plakken** (of dubbelklik op de keuzelijst) en klik op **OK**

- Sluit de query af, en bewaar de wijzigingen

Als je de keuzelijst uitprobeert, zul je in eerste instantie merken dat het prima werkt; de lijst laat de landen zien van het geselecteerde land uit de keuzelijst cboLanden. Maar als je vervolgens een ander land selecteert, zie je nog steeds de locaties uit het eerste land, en dat is uiteraard niet de bedoeling! Je wilt bij het wijzigen van het land, wel de daarbij behorende locaties zien!

Dat de lijst niet wordt bijgewerkt, is een typische eigenschap van formulieren: ze laten altijd een *momentopname* zien. Zodra er iets verandert in het formulier, moet die verandering nadrukkelijk worden 'doorgegeven' aan objecten die van de verandering afhankelijk zijn. In dit voorbeeld werkt dat als volgt: zodra je in de keuzelijst met Landen een ander land kiest, moet de keuzelijst met Locaties 'verteld' worden dat er een ander land is geselecteerd, en dat het criterium van de keuzelijst dus is veranderd. We doen dat met de opdracht *Requery*. Het makkelijkst doen we dat op één van de twee volgende plekken: bij de gebeurtenis <*Na bijwerken*> van de keuzelijst Landen (we hebben immers een ander land geselecteerd) of bij de keuzelijst <*Bij kiezen*> van de keuzelijst Locaties (als we een locatie gaan kiezen, willen we de op dat moment het actuele land gebruiken).

Welke optie je gebruikt, maakt eigenlijk niet zoveel uit. Een overweging zou kunnen zijn, dat de gebeurtenis <Bij kiezen> bij elke verandering van locatie opnieuw wordt uitgevoerd, ook als er geen ander land is geselecteerd. Zet je de actie op de keuzelijst cboLanden, dan wordt de gebeurtenis alleen uitgevoerd als er een ander land is geselecteerd. In beginsel zou dat genoeg moeten zijn, om de tweede keuzelijst goed te laten werken. De code <Bij kiezen> ziet er zo uit:

```
Private Sub cboLocatie_Enter()
   Me.cboLocatie.Requery
End Sub
```

Zet je de code op de keuzelijst cboLanden, dan ziet de code er zo uit:

```
Private Sub cboLand_AfterUpdate()
   Me.Filter = "[LandNaam] = '" & Me.cboLand & "'"
   Me.FilterOn = True
   Me.cboLocatie.Requery
End Sub
```

Je ziet dat dezelfde code gebruikt wordt: de actie Requery op de keuzelijst cboLocatie.

Methode 2: de SQL van de keuzelijst baseren op de eerste keuzelijst

De omschrijving van de tweede methode is misschien een beetje cryptisch, maar ik bedoel er het volgende mee: de tweede keuzelijst krijgt zelf geen SQL mee, maar krijgt een rijbron toegewezen op basis van de *gekozen waarde* uit de eerste keuzelijst. We gaan de SQL dus letterlijk helemaal opbouwen, en als rijbron gebruiken.

Voor deze methode kunnen we weer gebruik maken van een trucje: eerst maak je de afhankelijke keuzelijst op basis van methode 1, en vervolgens kopieer je de nieuwe SQL van de keuzelijst en gebruik je die om een SQL opdracht te maken. Als ik de SQL code van de keuzelijst opvraag, krijg ik dit: (enigszins opgeschoond, maar perfect werkend…)

```
SELECT DISTINCT LocatieNaam FROM qDuikplaatsen
WHERE (LandNaam=[Forms]![fDuikplaatsen]![cboLand])
ORDER BY LocatieNaam;
```

In deze code moet de tweede regel worden aangepast; hierin wordt namelijk een vergelijking gemaakt met de *keuzelijst* <cboLand>. En we willen niet meer verwijzen naar de Keuzelijst, maar

naar de gekozen *Waarde*.

We hebben al eerder gezien dat we een waarde uit een keuzelijst kunnen opvragen met het commando *Object.Value*. Dat gaan we hier dus ook doen. We vervangen [Forms]![fDuikplaatsen]![cboLand]) derhalve door: Me.cboLand. Maar eerst moeten we een variabele declareren, want we gaan de SQL code toewijzen aan een tekstvariabele. De eerste regel na Private Sub wordt dan ook:

Private Sub cboLand_AfterUpdate()

Dim strSQL As String

De SQL-code die hierboven staat moet nu worden toegewezen aan deze variabele. Als we de code gekopieerd hebben uit de rijbron, en vervolgens plakken in de procedure <Na bijwerken> van de keuzelijst cboLand, dan ziet de code er zo uit:

```
SELECT DISTINCT LocatieNaam FROM qDuikplaatsen
WHERE (LandNaam = [Forms]![fDuikplaatsen]![cboLand])
ORDER BY qDuikplaatsen.LocatieNaam;
```

Aan de rode kleur van de regels zie je al, dat er iets niet helemaal jofel gaat: twee regels zijn op dit moment niet correct. En dat klopt natuurlijk ook, want de SQL is nog niet toegewezen aan onze variabele. Dat kun je veranderen, door de eerste regel toe te wijzen aan de variabele.

Typ voor de eerste regel: **strSQL="**
De eerste regel ziet er nu zo uit:

strSQL = "SELECT DISTINCT LocatieNaam FROM qDuikplaatsen"

Access plaatst zelf een dubbele quote achter de regel; een tekststring staat namelijk altijd tussen twee dubbele quotes.

De volgende twee regels moeten uiteraard ook aan de variabele worden toegewezen. Dat kan op twee manieren: je kunt de regel ‘uitbreiden’ met een regeleinde of je kunt elke regel apart als tekst ‘optellen’ bij de variabele. In het eerste geval gebruik je het Underscore teken (_) als regeleinde, en ziet de uiteindelijke code er zo uit:

strSQL = "SELECT DISTINCT LocatieNaam FROM qDuikplaatsen " _

& "WHERE (LandNaam = '" & Me.cboLand & "') " _

& "ORDER BY LocatieNaam;"

De tweede variant ziet er zo uit:

strSQL = "SELECT DISTINCT LocatieNaam FROM qDuikplaatsen "

strSQL = strSQL & "WHERE (LandNaam = '" & Me.cboLand & "') "

strSQL = strSQL & "ORDER BY LocatieNaam;"

Merk op dat het verschil vooral aan het begin van de regels staat: de eerste regel begint met strSQL = ", en de tweede en derde regel met: strSQL = strSQL & "

Er wordt dus een nieuwe regel toegevoegd aan een bestaande variabele. Deze techniek, het toevoegen van tekst of waarden aan een bestaande variabele zul je nog vaak tegenkomen. De eerste variant levert exact hetzelfde eindresultaat op, dus welke vorm je kiest mag je weer zelf bepalen!

Let ook op de toevoeging achter Me.CboLand, de tekst & "' ". We maken een filter op basis van een Tekstwaarde; dat houdt dus in dat het filter ook een tekstopmaak moet krijgen. En een tekststring in een query of filter zetten we tussen dubbele quotes, of enkele quotes.

Opmerking:
Omdat een dubbel aanhalingsteken ook al gebruikt wordt om het begin- en eindpunt van een tekststring te definiëren, zitten we eigenlijk met een probleem: hoe vertel je Access dat het dubbele

aanhalingsteken geen beginpunt of eindpunt is van de string, maar een onderdeel van het tekstfilter? We kunnen dat op twee manieren oplossen: om te beginnen door dus een enkel aanhalingsteken te gebruiken i.p.v. een dubbel. Maar het kan ook op de volgende manieren: je kunt het dubbele aanhalingsteken vervangen door de ASCII waarde te gebruiken (Chr(34)). Het resultaat is dan: **LandNaam = " & Chr(34) & Me.cboLand & Chr(34) & " "**.
Of je typt er een extra dubbel aanhanlingsteken voor. De code ziet er dan zo uit: **LandNaam = """ & Me.cboLand & """ "** .

Vanwege de leesbaarheid, gebruik ik daarom vaak het enkele aanhalingsteken. Het is simpel, en de code blijft goed leesbaar. Ook hier geldt: elke methode is prima, en je mag zelf bepalen welke variant je wilt gebruiken.

Let overigens ook nog op de spaties aan het eind van elke regel; als je tekstregels samenstelt moet je er op letten dat de woorden wel van elkaar gescheiden blijven. Dat houdt dus in dat je ofwel een regeleinde toevoegt aan je teksten, ofwel de regel met een spatie eindigt of begint.

De rijbron opnieuw instellen

Als (bijna) laatste aanpassing moeten we de nieuwe SQL nog toewijzen aan de keuzelijst. Dat doen we met de opdracht Object.RowSource. De regel luidt nu: Me.cboLocatie.RowSource = strSQL. De complete code ziet er uiteindelijk zo uit:

```
Private Sub cboLand_AfterUpdate()
Dim strSQL As String
   Me.Filter = "[LandNaam] = '" & Me.cboLand & "'"
   Me.FilterOn = True
   strSQL = "SELECT DISTINCT LocatieNaam FROM qDuikplaatsen " _
      & "WHERE (LandNaam = '" & Me.cboLand & "') " _
      & "ORDER BY LocatieNaam;"
   Me.cboLocatie.RowSource = strSQL
   Me.cboLocatie.Requery
End Sub
```

De eerste aanpassing was dus het vastleggen van de tekstvariabele strSQL. Vervolgens werd de SQL toegewezen aan de variabele, waarbij de waarde uit de keuzelijst cboLand als letterlijke tekst in het filter werd gezet. Daarna werd de tekstvariabele met de SQL toegewezen aan de keuzelijst cboLocatie. Omdat er een nieuwe rijbron is gemaakt voor de keuzelijst, moet ook hier weer een Requery worden gebruikt om de inhoud van de keuzelijst te verversen.

Een keuzelijst leegmaken bij kiezen

De finishing touch is het leegmaken van de keuzelijst Locatie als we de keuzelijst Landen activeren. Immers, als we een nieuw land kiezen, is de eerder gekozen locatie niet meer geldig; die hoort immers bij een ander land. Dus is het wel netjes om de keuzelijst Locatie leeg te maken. Een mooi moment om dat te doen, is bij het activeren van de keuzelijst Landen. Dat kan bij de gebeurtenis *<Bij Kiezen>* van de keuzelijst Landen. De code is vrij simpel, en bestaat uit één opdracht: maak de keuzelijst cboLocatie leeg. En ziet er zo uit:

```
Private Sub cboLand_Enter()
   Me.cboLocatie = ""
End Sub
```

Opdracht

Maak een keuzelijst met invoervak voor de *Duikplaatsen*, die afhankelijk is van de keuzelijst *Locaties*. Gebruik hiervoor een van de twee hierboven uitgelegde methodes.

Samenvatting

We zijn begonnen met het maken van Afhankelijke keuzelijsten. We zijn daarbij begonnen met een keuzelijst om een record op te zoeken in een formulier. Deze keuzelijst was gebaseerd op het resultaat van de Wizard Keuzelijsten. We hebben gezien hoe we de Rijbron van een keuzelijst kunnen aanpassen aan onze doeleinden. En zijn we begonnen met het maken van keuzelijsten met invoervak om een (doorlopend) formulier te filteren. Daarna hebben we behandeld hoe je een keuzelijst afhankelijk kan maken van andere keuzelijsten, zodat de eerste keuzelijst de inhoud van de volgende keuzelijst kan bepalen.

Volgende Aflevering

In dit hoofdstuk hebben we keuzelijsten gemaakt waarin één keuze gemaakt kan worden. Die zijn prima te gebruiken om formulieren te filteren op één waarde. Het komt echter ook voor je een formulier wilt filteren op basis van meerdere keuzes, zoals twee landen of twee plaatsen. In ons databasevoorbeeld zou je bijvoorbeeld een lijst willen zien van duikplaatsen in Nederland en België. Of je wilt kunnen filteren op delen van een tekst. Omdat de technieken daarvoor min of meer overeen komen, behandelen we dit onderwerp apart in een volgend hoofdstuk.