Cursus Access voor Beginners - Hoofdstuk 9

Handleiding van Helpmij.nl

Auteur: OctaFish

Oktober 2011

Formulieren: (Afhankelijke) keuzelijsten maken Deel 2

In het vorige hoofdstuk hebben zijn we begonnen met het maken van *Afhankelijke keuzelijsten*; we hebben daarbij alleen keuzelijsten gemaakt van het type *Keuzelijst met invoervak*. Deze keuzelijsten zijn snel te maken, en eenvoudig in het gebruik: je kunt met de keuzelijst één waarde selecteren, en die vervolgens gebruiken in een actie.

De andere manier om selecties te maken doen we dus met *Keuzelijsten*. Een keuzelijst is te beschouwen als een uitgeklapte keuzelijst met invoervak. Je ziet dus niet één optie, maar een aantal. Dat aantal wordt bepaald door de hoogte van de keuzelijst: hoe langer hij is, hoe meer opties je ziet. Daar waar de keuzelijst met invoervak dus één waarde laat zien (de gekozen waarde) kun je met een keuzelijst veel meer opties tonen. En selecteren.

Hoe maken we selecties met keuzelijsten?

Met het type *Keuzelijst* zijn de mogelijkheden om selecties te maken wat uitgebreider. Niet alleen kunnen we een gewone keuzelijst gebruiken om, net als de Keuzelijst met invoervak, één waarde te selecteren, maar je kunt een Keuzelijst ook gebruiken om meerdere waarden te selecteren. Dat selecteren kun je dan nog op twee manieren gebruiken, die we beide zullen meenemen in dit hoofdstuk.

Het belangrijkste probleem waar we tegen aan gaan lopen bij keuzelijsten is echter dat het *uitlezen* van de geselecteerde waarden totaal verschilt. En op zich is dat wel verklaarbaar; gebruik je de keuzelijst om één waarde te selecteren, dan is het voor Access vrij duidelijk wat er moet gebeuren: de optie die je aanklikt, is de gekozen waarde. En die wordt, net dus als de Keuzelijst met invoervak, toegewezen aan de eigenschap *Value*. Dit principe werkt dus niet op het moment dat je meer waarden kunt selecteren. Want hoe bepaal je dat je klaar bent met selecteren? Zodra je meer dan één optie mag aanklikken, is het moment waarop je klaar bent met selecteren niet meer te bepalen. Je kunt dat zelfs niet vastpinnen op het moment dat je alle waarden hebt aangeklikt, want misschien heb je er wel één teveel aangeklikt, en moet de selectie worden verminderd.

Wat we dan ook meestal doen, is de keuzelijst uitlezen op het moment dat we een andere actie triggeren. Dat kan het klikken op een knop zijn bijvoorbeeld. Of bij het kiezen van een ander tekstvak of keuzelijst met invoervak. Het belangrijkste is, dat de keuzelijst wordt uitgelezen *op het moment dat de waarden nodig zijn*.

In het vorige hoofdstuk hebben we een doorlopend formulier gemaakt om direct het resultaat te kunnen filteren. In dit hoofdstuk gaan we ongeveer hetzelfde doen, met dit verschil dat we voor het selectieformulier een onafhankelijk maken met afhankelijke keuzelijsten, en een los formulier dat de gekozen selectie laat zien op een apart doorlopend formulier.

Een onafhankelijk formulier met keuzelijsten maken

We beginnen met het maken van een Niet-afhankelijk formulier. Dat is dus een formulier dat niet is gekoppeld aan een tabel of query. Vervolgens maken we de eerste keuzelijst met de wizard.

1. Zorg ervoor dat de knop Wizard is geselecteerd, en klik op de knop **Keuzelijst**
   De wizard *Keuzelijsten* verschijnt.

2. Kies: *De waarden voor de keuzelijst zullen worden opgezocht in een tabel* en klik op **Volgende**
3. Selecteer de tabel *Duikplaatsen* en klik op **Volgende**
4. Selecteer de velden **DuikplaatsID**, **Naam** en **SoortID** en klik op **Volgende**
5. Sorteer op het veld *Naam* en klik op **Volgende**
6. Maak de kolom *SoortID* ongeveer 1 cm breed door de rechter kolomlijn naar links te slepen en klik op **Volgende**
7. Typ **Landen** in het veld *Welk label wilt u bij de keuzelijst?* en klik op **Voltooien**

Als je de keuzelijst gaat uitproberen, zullen er wel een aantal dingen opvallen. Om te beginnen: je ziet niet alleen de landen, maar alle records in de tabel. Verder is de lay-out niet geweldig, en kun je maar één item selecteren. Er is dus nog wat werk voor de boeg!

Om te beginnen: de lay-out van de keuzelijst. Zoals ik al eerder verkondigde, is bij een keuzelijst de grootte op het formulier de grootte waarmee de gebruiker moet gaan werken. Als je het object dus breder en langer maakt, komt die grootte ook op het gebruikersformulier te staan. Dus door met de object afmetingen te spelen, kun je de keuzelijst er al beter uit laten zien.

De keuzelijst instellen op meervoudige selectie

De belangrijkste eigenschap die je bij een keuzelijst kunt aanpassen, is het selectiegedrag. Wel of niet meer opties selecteren, dat is de kwestie! Zoals in de inleiding al gezegd, afhankelijk van deze eigenschap verandert de manier waarop we de keuzelijst uitlezen nogal drastisch.

Het aanpassen van deze eigenschap gebeurt op het tabblad <Overige> middels de optie <*Meervoudige selectie*>. Hier vind je drie opties: <Geen>, <*Enkelvoudig*> en <*Uitgebreid*>. Laat je niet in de luren leggen door de term *Enkelvoudig*; hier wordt niet mee bedoeld dat je maar één optie kunt selecteren. Net zo min als *Geen* betekent dat je niks kunt selecteren! (wat de keuzelijst overigens waardeloos zou maken…) De verschillende opties zijn als volgt:

- *Geen*
  Deze optie betekent, dat de eigenschap *Meervoudig* is uitgeschakeld. Het is dus niet mogelijk om meer dan één waarde te selecteren.
- *Enkelvoudig*
  Met deze optie is het selecteren voor de gebruiker het simpelst: elke waarde die wordt aangeklikt, is gelijk geselecteerd. Nogmaals op dezelfde waarde klikken, en de selectie is opgeheven.
- *Uitgebreid*
  Met deze optie lijkt het alsof de keuzelijst weer één keuze accepteert. Niets is minder waar: met de <Ctrl> toets ingedrukt kun je de selectie uitbreiden of verkleinen. En met de <Shift> toets ingedrukt selecteer je alle waarden tussen de eerste geselecteerde optie en de optie die wordt aangeklikt. Een beetje vergelijkbaar dus met de manier waarop in de Verkenner bestanden kunnen worden geselecteerd. Uiteraard kun je de selectie met <Ctrl> en <Shift> selecties combineren.

1. De keuzelijst die we daarnet hebben gemaakt, gaan we volgens de optie *Enkelvoudig* gebruiken. Selecteer dus de optie **Enkelvoudig** en klik op het tabblad *Gegevens*

2. Test de keuzelijst weer uit, en probeer eventueel ook de optie *Uitgebreid*. Zet de instelling weer terug op *Enkelvoudig*

De rijbron van de keuzelijst beperken tot landen

Het tweede probleem, de lijst terugbrengen tot alleen landen, moeten we aanpakken in de *Rijbron* van de keuzelijst. Het probleem is eigenlijk hetzelfde als bij het formulier met de Keuzelijsten met Invoervak, maar op dit formulier kiezen we een andere aanpak. Ik wil namelijk niet alleen alleen de namen van de landen zien, maar ook het aantal locaties dat per land beschikbaar is. En daarvoor moet de rijbron drastisch worden aangepakt.

Hoewel we prima kunnen werken met de tabelnamen zoals ze zijn, is het toch wel handiger om de tabellen een *Alias* te geven, zodat we ze beter uit elkaar kunnen halen als we ermee gaan werken.

10. Klik op de knop met de puntjes achter de *Rijbron*, en voeg de tabel *Duikplaatsen* nogmaals toe middels de knop *Tabel weergeven* ()
De tweede tabel krijgt een extensie in de naam: *st_DuikPlaatsen_1* bijvoorbeeld.

11. Klik op de knop *Eigenschappen* en geef de tabel *Duikplaatsen* de Alias naam **Lan** en *st_DuikPlaatsen_1* de naam **Loc.**

De twee tabellen zijn nog niet aan elkaar gekoppeld; dat moeten we uiteraard nog wel doen, om te voorkomen dat we een Cartesisch product maken (zie eerder in de cursus, als je niet meer weet wat dat is).

Een tabel die afhankelijk is van zichzelf…

Om te weten hoe je de tabellen moet koppelen, en waarom je überhaupt een tabel aan zichzelf zou willen koppelen, moeten we eerst de structuur en gegevens van de tabel nog eens nader bestuderen.

Vaak zie je dat voor verschillende categorieën eigen tabellen worden gemaakt, wat op zichzelf overigens niet verkeerd hoeft te zijn; als je van een Land compleet andere informatie wilt opslaan als voor je duikplaatsen, dan zou je zo'n constructie prima kunnen overwegen. Je systeem is dan echter wel afhankelijk van de vastgelegde categorieën en subcategorieën; voor elk niveau moet je dan een aparte tabel maken, die dan vermoedelijk ook weer een apart formulier nodig heeft, eigen queries etc.

Terug naar onze tabel…

DuikPlaatsenId	ParentID	SoortID	Naam
1		1	Aruba
2		1	Egypte
3		1	Filipijnen
4		1	Nederland
5	1	2	Caribische Zee
6	3	2	Cebu
7	2	2	Dahab
8	2	2	El Gouna
9	4	2	Grevelingen
10	2	2	Hurghada
11	2	2	Marsa Alam
12	4	2	Oosterschelde
13	3	2	Sabang, Puerto Galera
14	2	2	Safaga

Zoals je ziet in de afbeelding, heeft het land *Filipijnen* de sleutelwaarde 3 in het veld *DuiplaatsenID*. Het veld *ParentID* is bij dit land leeg, net als bij alle andere landen. Verder zie je de waarde 1 staan in het veld SoortID. In de bijbehorende tabel zie je dat daarmee de kwalificatie *Land* wordt aangeduid.

We hadden al eerder gezien dat de landen verschillende locaties (kunnen) hebben, en dat bij de Filipijnen twee locaties horen: *Cebu* en *Sabang*. Deze records zie je ook terug in de afbeelding, resp. als DuikplaatsID 6 en DuikplaatsID 13. Beide records hebben de waarde 3 in het veld *ParentID*, en de waarde 2 bij *SoortID*. En dat staat weer voor *Locatie*. Blijkbaar verwijst de waarde in *ParentID* naar een record dat hoger in de rangorde staat.

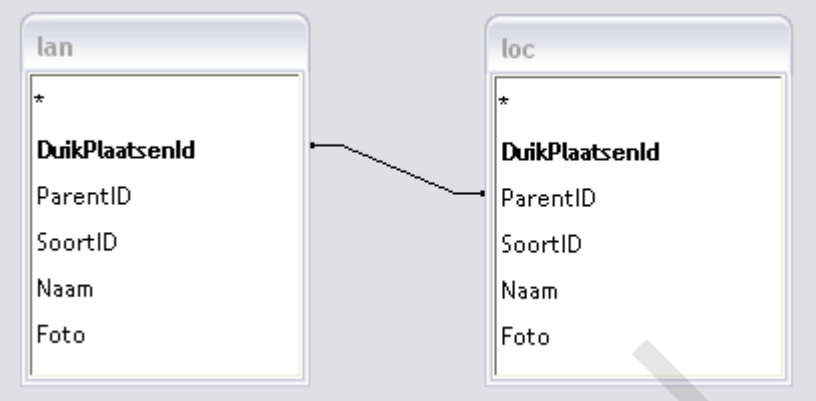
Hetzelfde geldt voor de feitelijke Duikplaatsen; deze krijgen een ParentID die verwijst naar de Locatie waar ze gevonden kunnen worden. Zo is de duikplaats *Sunken Island* te vinden op de locatie *Cebu. Het DuikplaatsID* van dit record is 255, het ingevulde *ParentID* is 6 (ga dit na) en het *SoortID* is 3 (controleer dit zelf).

Het grote voordeel van deze manier van werken, met verwijzingen naar een hoger niveau middels een kolom ParentID, is dat je een oneindig aantal niveau's kunt gebruiken in één tabel; het maakt niet uit of je met twee niveau's werkt, of met 12: bij elk nieuw record kun je opgeven onder welk record het nieuwe record komt te hangen. Hangt er aan een bepaald record nog niks, dan wordt automatisch een extra laag toegevoegd aan de structuur. Deze manier van werken is dus erg handig voor bijvoorbeeld stambomen; je begint daarbij met twee personen die met elkaar een gezin stichten. Vervolgens stichten de kinderen een eigen gezin, wat dus een nieuw niveau toevoegt. De kinderen daarvan kunnen ook weer een gezin stichten, met eigen kinderen: wederom komt er een niveau bij… etc.

Opmerking
De kolom Soortid is in dit soort constructies overigens eigenlijk niet nodig, maar toegevoegd voor het gebruikersgemak. Het is nu namelijk heel makkelijk om te bepalen hoeveel niveau's er in de tabel gebruikt worden; immers: voor elk niveau gebruiken we een oplopend getal. Het hoogste getal is dus automatisch het aantal niveau's dat in gebruik is.

Om de tabellen te kunnen koppelen, moeten we uiteraard weten wel veld we op wel veld moeten zetten. Dat antwoord is eigenlijk simpel: het DuikplaatsenID moet overeenkomen met

het ParentID van de locatie, dus je moet uit Lan de koppeling maken van het veld DuikplaatsenID op het veld ParentID uit de tabel Loc. Dat ziet er dan zo uit:



Als je de query uitvoert, zie je een resultaat dat toch niet helemaal voldoet: We zien niet alleen landen, maar ook locaties in de lijst. Eigenlijk wel verklaarbaar: we hebben een koppeling tussen DuikplaatsID en ParentID, maar nog niet gespecificeerd op welk niveau die relatie moet worden getoond. We zien dus niet alleen de koppelingen tussen Landen en Locaties, maar ook die tussen Locaties en Plaatsen! De lijst moet dus nog een extra restrictie krijgen, en hier komt het veld SoortID van pas: we willen namelijk alleen de Landen zien, en dat houdt in dat de SoortID dus de waarde 1 moet hebben. We kunnen dit simpel als criterium toevoegen. Bovendien willen we ook het *aantal records* weten, dus we hebben ook nog een veld nodig dat die berekening uitvoert

DuikPlaatsenId	Naam
1	Aruba
5	Caribische Zee
6	Cebu
7	Dahab
2	Egypte
8	El Gouna
3	Filipijnen
9	Grevelingen
10	Hurghada
11	Marsa Alam
4	Nederland
12	Oosterschelde
13	Sabang, Puerto Galera
14	Safaga
15	Sharm-El-Sheikh

12. Ga terug naar het Ontwerpscherm van de query, en voeg het veld *SoortID* toe
13. Typ in de regel *Criteria* de waarde **1** en kies in de regel *Totaal* de waarde **Waar**
14. Sleep het veld *DuikplaatsenID* uit de tabel *Loc* naar het raster, en kies in de regel *Totaal* de waarde **Aantal.** Typ vóór de veldnaam de tekst **Aantal:**
15. Voer de query nogmaals uit

Het resultaat is nu de lijst met landen, plus het aantal locaties.

DuikPlaatsenId	Naam	Aantal
1	Aruba	1
2	Egypte	6
3	Filipijnen	2
4	Nederland	2

Zoals je kunt zien, wordt netjes voor elk land aangegeven hoeveel locaties er zijn.

16. Om duidelijk te maken waar de tweede kolom voor staat, kunnen we de optie <*Kolomkoppen*> op **Ja** zetten.

De keuzelijst is nu af, en hij doet precies wat we verwachten: we zien de landen, met het aantal locaties, en we kunnen meer dan één land selecteren. We kunnen dus aan de volgende keuzelijst gaan beginnen!

17. De laatste stap is om de keuzelijst een goede naam te geven. Normaal gesproken zou ik de lijst *lstLanden* noemen; omdat dit formulier straks functionaliteit gaat krijgen die eist dat de keuzelijsten een uniforme naam krijgen, krijgt deze keuzelijst de naam **lstCat1**

Een keuzelijst afhankelijk maken van een andere keuzelijst

Bij de tweede keuzelijst lopen we gelijk tegen een probleem op: de keuzelijst Locatie moet namelijk alle locaties laten zien van de landen die we selecteren in de eerste lijst.

We hadden al geconstateerd dat we de keuzelijst op een andere manier moeten ‘uitlezen’, omdat het niet meer mogelijk om de eigenschap *Value* te gebruiken. Daarom zal ik de code regel voor regel opbouwen, en uitleggen. De code moet worden gemaakt op basis van de eigenschap <Na bijwerken> van de keuzelijst Landen. De verandering in die lijst bepaalt namelijk wat er in de tweede lijst moet worden getoond. We openen dus eerst de gebeurtenis <Na bijwerken> van de keuzelijst *lstCat1*.

- **opmerking**
  Om onderscheid te houden tussen Keuzelijsten met Invoervak, en Keuzelijsten gebruik ik voor een Keuzelijst de naam lstKeuzelijst, en voor een Keuzelijst met Invoervak de naam cboKeuzelijst. De naamgeving is uiteraard facultatief; je mag daar je eigen namen voor gebruiken.

18. Maak een gebeurtenis op de eigenschap <*Na bijwerken*> van de keuzelijst *lstCat1*

Eerst gaan we wat variabelen vastleggen; omdat we dezelfde variabelen op meerdere plekken gaan gebruiken, is het zinvol om ze in het Algemene deel van de procedure vast te leggen; dat is dus onder de regel Option Compare Database.

19. Verplaats de cursor naar het bovenste deel van de module, en typ de volgende variabelen:

```
Dim strSQL As String, sCat As String
Dim i As Integer, iAantal As Integer
Dim itm As Variant
Dim ctl As Control
```

```
Const iHoog = 295
Const iRijHoog = 23
```

20. Plaats de cursor weer in de routine *Private Sub lstCat1_AfterUpdate()*

21. Typ onderstaande tekst over:

```
strSQL = "SELECT p.DuikPlaatsenId, p.Naam, Count(p.DuikPlaatsenId) AS Aantal,
p.ParentID " _
      & "FROM st_Duikplaatsen AS p LEFT JOIN st_Duikplaatsen AS l ON
p.DuikPlaatsenId = l.ParentID"

   strSQL = strSQL & vbCrLf & "WHERE "
```

Dit is het algemene deel van de SQL die we nodig hebben voor de tweede keuzelijst. Normaal gesproken zouden we achter WHERE zetten: *p.ParentID =" & Me.lstCat1.Value & " "*

Maar ik heb al gezegd dat dit niet gaat werken. We zullen de keuzelijst regel voor regel moeten uitlezen. We doen dat met de eigenschap < *ItemsSelected*>. Deze eigenschap vinden we in de groep met eigenschappen van de keuzelijst. De actie die we gaan maken leest elke waarde uit de keuzelijst uit, en als de eigenschap ItemsSelected waar is, voegen we de waarde toe aan de filterstring.

Stap 1 is bepalen in wat voor variabele we de keuzelijst gaan uitlezen. Normaal gesproken zou je dat in een tekststring willen doen. Jammer genoeg kan dat niet; Access ondersteunt alleen variabelen van het type *Variant*. We hebben daar één variabele voor vastgelegd: de variabele *itm*. De uitgelezen waarden zetten we wèl in een tekststring, de variabele *sCat*. Die maken we eerst leeg, voordat we met het uitlezen beginnen. Ook hebben we een teller nodig, om de string correct op te bouwen. Die geven we een beginwaarde van Nul.

22. Typ de volgende regels:

```
sCat = ""i = 0
  For Each itm In Me.lstCat1.ItemsSelected
    If Me.lstCat1.ItemsSelected.Count > 0 Then
```

De routine begint feitelijk met de regel <*For Each itm In Me.lstCat1.ItemsSelected*>. Wat hier gebeurt is het volgende: met <*For each itm*> lopen we door elk geselecteerd item dat is geselecteerd door de gebruiker. Die selectie is vastgelegd in de eigenschap <*ItemsSelected*>. Vervolgens starten we een IF…THEN…ELSE…END IF routine. Deze is nodig om te checken of er wel een selectie is gemaakt. Zo niet, dan heeft het uiteraard geen zin om een filter te maken.

23. Typ de volgende regels:

```
sCat = sCat & "(p.ParentID = " & Me.lstCat1.ItemData(itm) & ") "
    i = i + 1
      If i < Me.lstCat1.ItemsSelected.Count Then sCat = sCat & "OR "
```

In dit deel wordt het feitelijke filter opgebouwd. Je ziet misschien waarom we zijn begonnen met het leegmaken van het filter; met dit deel van de opdracht: *<sCat = sCat & "">* wordt bij elk item de variabele sCat uitgebreid; omdat we wel met een lege string willen beginnen, wordt hij dus eerst leeg gemaakt voordat we de keuzelijst uitlezen.

Ook zie je dat we in bijna alle gevallen het woord OR toevoegen aan de variabele; net als bij het zelf intypen van een criterium in een query, moet je de verschillende waarden scheiden, om het filter goed te laten werken. Als je bijvoorbeeld twee landen selecteert, dan is de locatie die je wilt zien gelegen in één van de twee geselecteerde landen: Ofwel land 1, ofwel land 2. In een query scheid je die opties met het gereserveerde woord OR. In het filter moet dat woord dus worden toegevoegd bij een selectie van meer dan één.

De regel *<i = i + 1>* verhoogt bij elk geselecteerd item een teller met de waarde 1 De teller wordt vervolgens vergeleken met het totaal aantal geselecteerde items:
<If i < Me.lstCat1.ItemsSelected.Count>.
Deze regel is nodig om de verschillende elementen in het filter te kunnen scheiden, en om te voorkomen dat bij het laatste item ook nog het woord OR wordt toegevoegd.

Dat principe is vrij makkelijk te snappen, als je als voorbeeld één item selecteert. Het filter bestaat dan uit één waarde, en het woord OR is dan niet nodig. Pas bij twee landen (of meer) moet je de verschillende landen gaan scheiden; alleen na het laatste land hoeft het woord OR niet meer te worden toegevoegd. Vandaar de teller die steeds bij elk item wordt opgehoogd. En de vergelijking met het totale aantal geselecteerde items: zolang de teller kleiner is dan het aantal geselecteerde waarden, voegen we het woord OR toe. Is de teller gelijk aan het aantal items, dan weten we dat we bij het laatste item zijn aanbeland.

We maken de routine af met de volgende regels.

24. Typ de volgende regels:

```
Else
  Exit Sub
   End If
    Next itm
```

Deze regels zijn op zich niet zo spannend; de ELSE hoort bij de IF waarin we controleren of er wel iets is geselecteerd. Is dat niet het geval, dan gaat de routine naar het ELSE deel, en dat zegt alleen maar: stoppen met de routine!

De regel Next itm sluit vervolgens de FOR EACH loop af; de regel zegt: ga naar het volgende geselecteerde item.

Nu de variabele sCat is opgebouwd, is het tijd om hem toe te voegen aan de SQL string die uiteindelijk de lijst moet gaan opbouwen.

25. Typ de volgende tekst over:

```
sCat = sCat & vbCrLf
 strSQL = strSQL & sCat
```

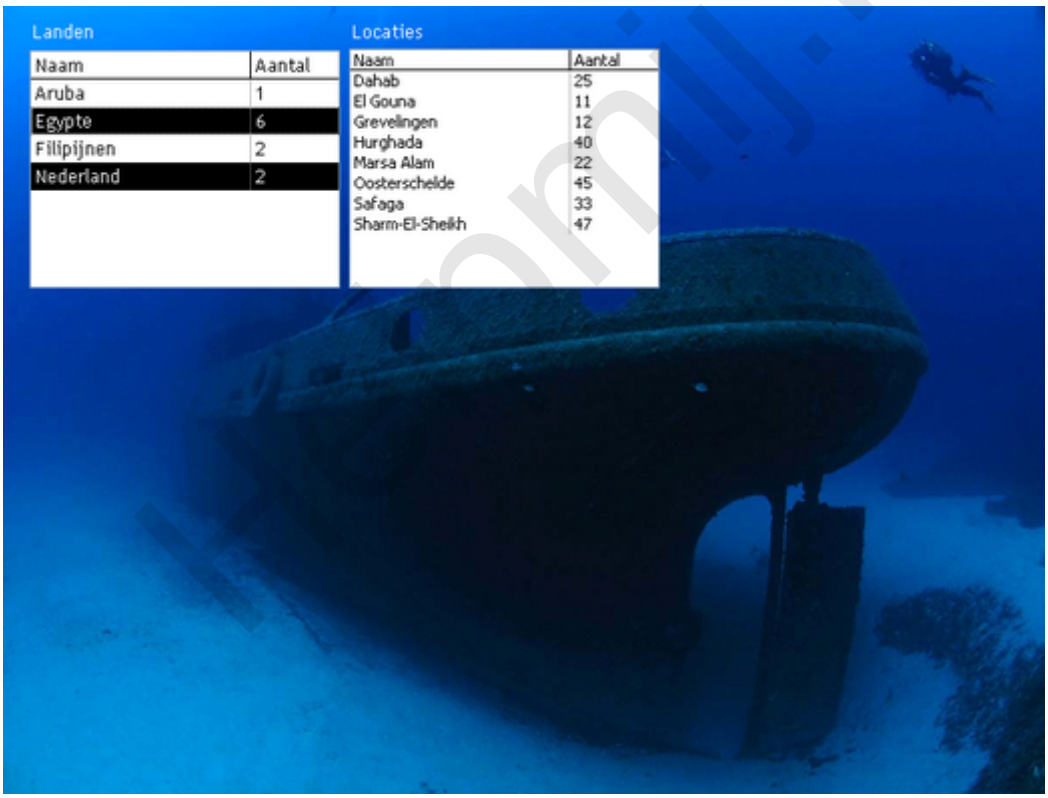
strSQL = strSQL & "GROUP BY p.DuikPlaatsenId, p.Naam, p.ParentID" & vbCrLf _
& "ORDER BY p.Naam"

Eerst wordt een lege regel toegevoegd aan de variabele. Dit mag ook een spatie zijn, zolang we er maar voor zorgen dat het woord GROUP vrij staat, en niet tegen het laatste woord van de variabele aan wordt geplakt. Je krijgt dan namelijk een foutmelding. Vervolgens wordt de variabele *sCat* toegevoegd aan de variabele strSQL. Als volgende regel wordt het Groepeerniveau toegevoegd, en het veld waarop we gaan sorteren.

26. Typ de volgende twee regels:

Me.lstCat2.RowSource = strSQL
Me.lstCat2.Requery

Hiermee wordt de SQL toegewezen aan de tweede keuzelijst. Je ziet, dat de naam van de keuzelijst identiek is aan de eerste keuzelijst, met slechts een ander nummer. De reden daarvan wordt later behandeld… Het formulier ziet er nu zo uit:



En de totale code is als volgt:

Private Sub lstCat1_AfterUpdate()

strSQL = "SELECT p.DuikPlaatsenId, p.Naam, Count(p.DuikPlaatsenId) AS Aantal,p.ParentID " _
& "FROM st_Duikplaatsen AS p LEFT JOIN st_Duikplaatsen AS l ON p.DuikPlaatsenId = l.ParentID"
strSQL = strSQL & vbCrLf & "WHERE "

```
    sCat = ""
    i = 0
    For Each itm In Me.lstCat1.ItemsSelected
       If Me.lstCat1.ItemsSelected.Count > 0 Then
          sCat = sCat & "(p.ParentID = " & Me.lstCat1.ItemData(itm) & ") "
          i = i + 1
          If i < Me.lstCat1.ItemsSelected.Count Then sCat = sCat & "OR "
       Else
          Exit Sub
       End If
    Next itm

    sCat = sCat & vbCrLf
    strSQL = strSQL & sCat & "GROUP BY p.DuikPlaatsenId, p.Naam, p.ParentID" &
    bCrLf & "ORDER BY p.Naam"

    Me.lstCat2.RowSource = strSQL    Me.lstCat2.Requery
End Sub
```

Probeer de code uit, nadat je uiteraard wel hebt gecontroleer of je a) wel een tweede keuzelijst hebt gemaakt, en b) of de naam inderdaad *lstCat2* is.

Opdracht

Maak nu een keuzelijst voor de *Duikplaatsen*, die afhankelijk is van de keuzelijst *Locaties*. Gebruik hiervoor hetzelfde principe als voor de keuzelijst lstCat2. Noem de keuzelijst: lstCat3.

De laatste loodjes…. de keuzelijsten opmaken

De derde keuzelijst werkt ongeveer op dezelfde manier als de eerste, zoals je vermoedelijk al hebt gemerkt. Op zich wel verklaarbaar: elk record heeft namelijk maar maximaal één parentID, dus eigenlijk maakt het niet uit op welk niveau je de combinatie maakt: zodra je selecteert op een ParentID, krijg je de daaraan gerelateerde records. Dat is dus een mooie eigenschap van deze werkwijze!

Een nadeel van keuzelijsten is dat ze nogal veel plek in beslag nemen. Het zou fraai zijn, als we de hoogte van de keuzelijst kunnen aanpassen aan het aantal records dat erin getoond moet worden. Gelukkig kunnen we dat wel ongeveer berekenen, en dat gaan we in deze paragraaf dus laten zien. Verder is het wel zo logisch dat bij het leegmaken van een bepaalde keuzelijst, de daaronder hangende keuzelijsten ook leeg zijn. Bij de keuzelijst Locaties praten we dan over het leegmaken van Duikplaatsen, en bij de keuzelijst Landen is het logisch dat zowel Locaties als Duikplaatsen leeg is. En dus ook de standaardhoogte krijgen.

Om e.e.a. simpel in te stellen, is het eigenlijk een voorwaarde dat er een vaste structuur in de keuzelijsten zit. Vandaar dat ik de keuzelijsten de namen lstCat1, lstCat2 en lstCat3 heb gegeven. Op basis van deze namen is het relatief eenvoudig om een lus te maken die door keuzelijsten instelt.

Om te beginnen, moeten de variabelen worden uitgebreid. De volledige lijst aan variabelen wordt nu:

```
Dim strSQL As String, sCat As String
Dim i As Integer, iAantal As Integer
Dim itm As Variant
Dim ctl As Control
Const iHoog = 295
Const iRijHoog = 25
```

Je ziet dat er twee variabelen van het type *Constante* zijn bijgekomen: *iHoog* en *iRijHoog*. Deze variabelen worden gebruikt om de basisregelhoogte in te stellen (iHoog) en het aantal rijen dat zonder schuifbalk in een lijst kan worden getoond. Die schuifbalk is afhankelijk van de hoogte van de lijst, en het aantal records dat er in past. En uiteindelijk is de hoogte van het formulier daarin leidend; de keuzelijst moet uiteraard niet langer zijn dan het formulier hoog is.

Er komt een nieuw stukje code bij, die moet worden gestart bij het laden van het formulier. Dat is namelijk het juiste moment om het formulier de eerste keer op te maken. De code ziet er zo uit:

```
Private Sub Form_Load()

   Me.Form.InsideHeight = 9000
    For i = 1 To 3
      Me("lstCat" & i) = ""
       Me("lstCat" & i).Height = iHoog
   Next i

   strSQL = "SELECT l.DuikPlaatsenId, l.Naam, Count(l.DuikPlaatsenId) AS Aantal "
_
      & "FROM st_Duikplaatsen AS l LEFT JOIN st_Duikplaatsen AS p ON
     .DuikPlaatsenId=p.ParentID " _
      & " WHERE (((p.DuikPlaatsenId)>0) AND ((l.ParentID) Is Null)) GROUP BY
     .DuikPlaatsenId, l.Naam ORDER BY l.Naam;"

   With CurrentDb.OpenRecordset(strSQL)
      If .RecordCount > 0 Then
         .MoveLast
         .MoveFirst
         iAantal = .RecordCount
      Else
         iAantal = 1
      End If
      .Close
   End With

   Me.lstCat1.Height = iAantal * iHoog + iHoog

End Sub
```

Laten we daar het eerste stuk eens van onder de loep nemen:

```
Me.Form.InsideHeight = 9000
For i = 1 To 3
   Me("lstCat" & i) = ""
   Me("lstCat" & i).Height = iHoog
Next i
```

We beginnen met het vaststellen van de feitelijke hoogte van het formulier met de opdracht Me.Form.InsideHeight. De maatvoering daarvan is niet heel gebruikelijk, want de hoogte wordt aangeduid in Twips (1/20 van een punt of 1/1.440 inch. Er gaan 567 twips in een centimeter.)

Variabele namen opbouwen met een lus

Vervolgens maken we een lus die drie keer doorlopen wordt (For i=1 To 3) en waarbij de naam van elke keuzelijst wordt opgebouwd op basis van de waarde van i. Normaal gesproken verwijs je naar een object door de letters ME. (punt) in te typen, en dan de eerste letters van het object. Access laat dan een lijst zien van de objecten die met de lettercombinatie beginnen. Voor de drie keuzelijsten krijg je dan resp. Me.lstCat1, Me.lstCat2 en Me.lstCat3. Dit systeem werkt prima als je één object wilt aanpassen, maar het is niet mogelijk om deze lijst op die manier te genereren. Gelukkig kan dat wel als we de *naam* van het object kunnen samenstellen. En dat kan dus prima als we die naam generiek maken, en op basis van getallen uniek maken. En dat is precies wat we hier gedaan hebben: elke lijstnaam is, hoewel grotendeels identiek, toch uniek dank zij het volgnummer.

De naam wordt nu als volgt opgebouwd: je typt eerst de letters Me, daarna een Haakje openen, gevolgd door het generieke deel van de naam tussen dubbele aanhalingstekens. Oftewel: **Me("lstCat"**. De punt achter ME is dus vervangen door een haakje. Daarna voegen we het volgnummer toe en sluiten we het geheel af met een sluithaak: **& i)**.
En nu helemaal: **Me("lstCat" & i)**.

De rest van de opdracht zou niet meer vreemd moeten zijn: de lijst wordt eerst leeggemaakt met de opdracht ="" en op de volgende regel krijgt elke lijst de standaardhoogte. Die komt overeen met één regel tekst. Dus ongeveer de hoogte van een Keuzelijst met Invoervak.

De hoogte van een keuzelijst bepalen a.d.h.v. het aantal records

Nu de lijsten een minimale hoogte hebben gekregen bij het opstarten, moet de hoogte uiteraard nog worden herberekend als er in de bepalende keuzelijst iets is gekozen. Het zal niet verbazen dat we deze berekening toevoegen aan de gebeurtenis <Na bijwerken> van de keuzelijsten, die tenslotte al code bevatten om de nieuwe keuzelijst op te maken. Bij het opmaken van het filter komt er, om te beginnen, een regel bij:

```
sCat = sCat & "(p.ParentID = " & Me.lstCat1.ItemData(itm) & ") "
i = i + 1
iAantal = iAantal + lstCat1.Column(2, itm)
If i < Me.lstCat1.ItemsSelected.Count Then sCat = sCat & "OR "
```

We hebben een extra variabele gedefinieerd, en deze bevat de som van het aantal records dat we eerder in de query van de keuzelijst hadden gezet. Deze waarde bevat dus het aantal records dat we straks in de keuzelijst terug zien. De berekening van de hoogte van de keuzelijst kan vervolgens op elke plek na het doorlopen van de lus worden uitgevoerd; in mijn voorbeeld gebeurt het als laatste handeling.

We kijken eerst of het aantal rijen dat we nodig hebben groter is dan het aantal dat ik op basis van de constante iRijHoog heb vastgelegd. In het voorbeeld hanteer ik de waarde 25 als grens; bij minder dan 25 bepaalt het aantal records de hoogte, is het aantal records groter, dan maak ik de hoogte van de keuzelijst 25 records hoog. Deze berekening vindt weer plaats op basis van een IF…THEN…ELSE…END IF constructie.

```
If iAantal > iRijHoog Then ‘ de waarde 25 dus
   Me.lstCat2.Height = iHoog * iRijHoog + iHoog
Else
   Me.lstCat2.Height = iHoog * iAantal + iHoog
End If
```

Bij de uiteindelijke berekening tel ik nog een keer de waarde iHoog (de regelhoogte dus) op om extra ruimte te creëren voor de koptekst; laat je die weg, dan kan de berekening dus vermoedelijk worden aangepast.

- **opmerking**
  De gebruikte waarden zijn sterk afhankelijk van de beeldscherminstellingen, en vooral het gebruikte lettertype. Vermoedelijk zul je dus wat moeten variëren met de verschillende Constanten om de juiste berekening te maken, en dus ook de juiste hoogtes te krijgen. Het grote voordeel van het werken met constanten is, dat je maar op één plek de waarden hoeft te veranderen om de berekeningen aan te passen.

Samenvatting

We hebben in dit hoofdstuk Afhankelijke keuzelijsten gemaakt; we zijn daarbij begonnen met een keuzelijst om een land op te zoeken in een formulier, en daarna hebben we een keuzelijst gemaakt die de records lieten zien die bij de gekozen selectie horen. Keuzelijsten kunnen worden gebruikt om één waarde te selecteren, of meerdere. In het eerste geval is de methode van uitlezen identiek als bij een Keuzelijst met Invoervak. Bij de optie om meerdere waarden te selecteren verschilt de methode van uitlezen van de keuzelijst echter aanzienlijk. De methodiek die daar bij hoort stond centraal in dit hoofdstuk.

Ook hebben we gezien hoe we keuzelijsten konden opmaken, en de hoogte van de keuzelijsten konden laten afhangen van het aantal getoonde records.

Volgende Aflevering

In het volgende hoofdstuk maken we een begin met *Rapporten*. In de oudere versies van Access waren rapporten puur bedoeld om overzichten af te drukken; een functie die uiteraard nog steeds aanwezig is in de nieuwere versies. Omdat rapporten heel goed geschikt zijn om gegevens te ordenen en te presenteren, hebben de nieuwere versies een extra weergave methodiek gekregen die de gebruiker in staat stelt om de gegevens gegroepeerd te bekijken zonder ze te hoeven afdrukken.

We gaan dus ook wat nieuwe functies van Access 2010 bekijken, hoewel alles wat we gaan maken uiteraard ook in Access 2003 mogelijk is.