Cursus Access Voor Beginners - Hoofdstuk 13

Handleiding van Helpmij.nl

Auteur: OctaFish

Juli 2012

Een planningskalender maken

Een vraag die regelmatig terugkomt in het forum, is hoe je een kalender maandoverzicht kunt maken in Access met alle dagen in het overzicht, ongeacht of er gegevens zijn voor die dag. Die vraag is niet zo simpel te beantwoorden, omdat een database nu eenmaal werkt op basis van gegevens die in de tabel zijn opgeslagen. Je kunt makkelijk een overzicht maken van de dagen dat een persoon werkt, door in de tabel Planning te kijken welke dagen zijn ingevuld, maar zonder extra hulptabellen kun je niet laten zien op welke dagen die persoon er *niet* is. Daar heb je immers geen records voor. Hetzelfde probleem kom je tegen als je een Reserveringensysteem bouwt: je ziet wel wanneer een voorwerp is uitgeleend, maar niet de dagen dat het in huis is, en dus beschikbaar voor nieuwe verhuur.

We gaan in dit hoofdstuk een planningsrapport maken waarin we de bezettingsgraad van een kinderdagverblijf grafisch op een rapport weergeven. Het eindresultaat ziet er ongeveer zo uit:

In dit overzicht zie je per maand hoeveel kinderen er in de ochten aanwezig zijn, en hoeveel in de middag. Aan de hand van dit rapport is het dus eenvoudig om te bepalen hoeveel kinderen er nog bij kunnen, en hoeveel personeel je moet inzetten op een bepaald dagdeel van een bepaalde dag.

Als extraatje maken we het rapport met kolommen die zich in hoogte aanpassen aan de hoeveelheid personen, zodat je ook zonder de getallen kunt zien waar hoe de gegevens zich verhouden tot elkaar. Zo zie je in het voorbeeld heel snel dat de maandag ochtend beter bezet is als de vrijdag ochtend.

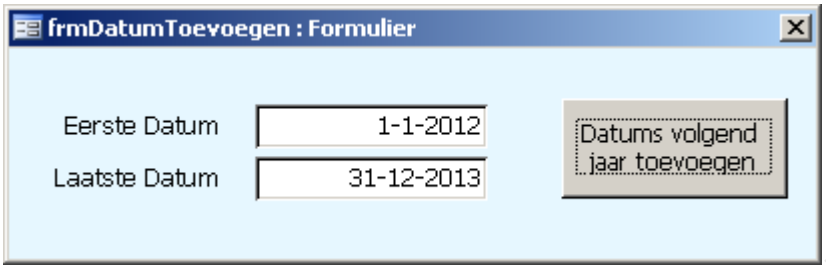
De voorwaarde: een hulptabel

Om zo'n rapport te kunnen maken, hebben we een tabel nodig met daarin de dagen die we terug willen zien in het overzicht. Die tabel heeft wat mij betreft twee varianten: ofwel een tabel met alle dagen van de week, ofwel een tabel met alleen werkdagen. Die laatste variant gebruik je bijvoorbeeld voor een personeelsplanning waarbij in het weekend niet gewerkt wordt. Dan hoef je die dagen uiteraard ook niet in het overzicht te zien.
De tabel die we op deze manier maken, gebruiken we later in een query die we gaan opzetten als een *Cartesisch product*. Deze term is in een van de eerste hoofdstukken ook al gevallen. Om het geheugen even op te frissen: dit is een query waarbij van een aantal tabellen alle mogelijke combinaties worden gemaakt. Heb je in tabelA 5 records, en in tabelB 12, dan zie je in het Cartesisch product van die twee 5 * 12 = 60 records terug.

Door in die query een filter aan te maken, hou je alleen de relevante combinaties over. En dat is dus wat we gaan doen met de Datum tabel.

De basisstappen

We hebben dus een extra tabel nodig met datums. Nu kun je een (arme) werkstudent inhuren en die een uurtje of wat datums laten inkloppen, maar dat kan natuurlijk veel simpeler, als we daar een functie voor gebruiken. Die functie kun je aanroepen vanaf een knop op een formulier bijvoorbeeld.



Dit formulier kijkt naar de tabel met datums, en laat de eerste datum zien, en de laatste. De eerste is in dit geval niet interessant, maar de laatste dus wel, want je wilt als je de nieuwe datums toevoegt natuurlijk niet dat een datum er twee keer in komt te staan: de eerstvolgende nieuwe datum moet dan ook aansluiten bij de laatste.

Laten we eens kijken wat de knop precies doet.

```
Function DatumsToevoegen()

Dim iBegin As Long, iStart As Long, iEind As Long, iLaatsteDag As Long

Dim dtEind As Date

Dim iDD As Integer

Dim sDD(2) As String

    With CurrentDb.OpenRecordset("SELECT DISTINCT Max(CLng([Dag])) " _

        & "AS tmp FROM Datums;")

        On Error Resume Next

        If .RecordCount = 1 Then

            iLaatsteDag = .Fields("tmp").Value

        End If

        .Close

    End With
```

De functie begint (uiteraard zou ik ondertussen mogen zeggen) met het declareren van de variabelen die we nodig hebben. De matrixvariabele sDD heeft een vaste dimensie van twee; hierbij is in de algemene sectie wel Option Explicit 1 gedeclareerd; laat je deze regel weg, dan begint de matrix bij 0, en gebruik je de declaratie sDD(1) om twee dimensies vast te leggen.

Vervolgens wordt in een Recordset de hoogste waarde uit de tabel [Datums] opgehaald en toegewezen aan de variabele iLaatsteDag. Deze waarde wordt overigens gelijk al vertaald naar een

getal, want we gaan de nieuwe datums baseren op de numerieke waarde van de datum. Vandaar dus Max(CLng([Dag])) als veld.

Daarna gaan we de begin- en einddatums bepalen die we willen toevoegen. Om het allemaal niet te ingewikkeld te maken, voegen we de datums per jaar toe. We hebben dus een startdatum nodig, en een einddatum. Deze worden ook weer aan variabelen toegewezen: iStart en iEind. In de volgende regel kijken we of de startdatum wel overeenkomt met de hoogste datum in de tabel (toegewezen aan de variabele iLaatsteDag). Voor het gemak behandel ik hier de variant die werkdagen aan de tabel toevoegt. Wil je alle dagen van de week, dan kun je de Do Until Weekday … Loop regels verwijderen. Deze lus controleert welke dag van de week de nieuwe dag is. Valt die op een zaterdag of zondag, dan moet de datum worden verhoogd, net zo lang tot het weer maandag is. Dan wordt ook de variabele iEind weer opnieuw gedefinieerd, want de startdatum zou nu in een volgend jaar kunnen liggen.

De Start- en Einddatum worden samengesteld met de functie DateSerial, waarin je op basis van een Jaartal, een Maandwaarde en een Dagwaarde een datum maakt.

```
iStart = Clng(DateSerial(Year(Date), 1, 1))

If iLaatsteDag > iStart Then

    iStart = iLaatsteDag + 1

    If Weekday(CDate(iStart), vbMonday) > 5 Then

        Do Until Weekday(CDate(iStart), vbMonday) = 1

            iStart = iStart + 1

        Loop

    End If

End If

iEind = CLng(DateSerial(Year(CDate(iStart)) + 1, 1, 1))
```

Vervolgens begint het echte werk: het vullen van de tabel. Hiervoor wordt de tabel [Datums] geopend met een Recordset, en de records toegevoegd met .AddNew en .Update. Daarbij wordt het getal (we werken immers met getallen) terugvertaald naar een datum met de functie CDate.

Dit alles vindt plaats in een lus, die net zolang doorgaat tot de toe te voegen datum gelijk is aan de einddatum. Ook hier weer de variant met werkdagen, te herkennen aan de lus If Weekday(CDate(iStart), vbMonday) < 6. Wil je ook de zaterdagen, dan maak je er van: < 7, wil je alle dagen, dan kun je de lus verwijderen.

We gebruiken bovendien nog een extra lus om ook de dagdelen toe te voegen; elke dag heeft in het rapport een ochtendbezetting, en een middagbezetting, en die dagdelen moeten dus ook in de basistabel voorkomen. Hiervoor gebruiken we een vaste matrix variabele, die we vullen met de twee codes.

De matrix wordt als volgt gedeclareerd:

```
Dim sDD(2) As String
```

En we vullen hem als volgt:

```
    sDD(1) = "VM"
    sDD(2) = "NM"
    iBegin = iStart
    With CurrentDb.OpenRecordset("Datums")
        Do While iStart < iEind
            For iDD = 1 To 2
            ''If Weekday(CDate(iStart), vbMonday) < 6 Then
                .AddNew
                !Dag = CDate(iStart)
                !DagDeel = sDD(iDD)
                .Update
            ''End If
            Next iDD
            iStart = iStart + 1
        Loop
        .Close
    End With
    MsgBox "Datums " & CDate(iBegin) & " - " _
        & CDate(iEind - 1) & " toegevoegd."
End Function
```

De volledige code:

```
Dim iBegin As Long, iStart As Long, iEind As Long, iLaatsteDag As Long
Dim dtEind As Date
Dim iDD As Integer
Dim sDD(2) As String

    With CurrentDb.OpenRecordset("SELECT DISTINCT Max(CDbl([Dag])) AS
```

```
LaatsteDag FROM Datums;")
        On Error Resume Next
        If .RecordCount = 1 Then
            iLaatsteDag = .Fields(0).Value
        End If
        .Close
    End With
    iStart = CDbl(DateSerial(Year(Date), 1, 1))
    iEind = CDbl(DateSerial(Year(Date) + 1, 1, 1))
    If iLaatsteDag > iStart Then
        iStart = iLaatsteDag + 1
        iEind = CDbl(DateSerial(Year(CDate(iStart)) + 1, 1, 1))
    End If

    sDD(1) = "VM"
    sDD(2) = "NM"
    iBegin = iStart
    With CurrentDb.OpenRecordset("Datums")
        Do While iStart < iEind
            For iDD = 1 To 2
            ''If Weekday(CDate(iStart), vbMonday) < 6 Then
                .AddNew
                !Dag = CDate(iStart)
                !DagDeel = sDD(iDD)
                .Update
            ''End If
            Next iDD
```

```
            iStart = iStart + 1

        Loop

        .Close

    End With

    MsgBox "Datums " & CDate(iBegin) & " - " & CDate(iEind - 1) & "
toegevoegd."

End Function
```

Vervolgens wordt de tabel gesloten, en krijg je nog een samenvatting in een dialoogvenster.

Opzet van de query

Voordat we de definitieve query kunnen maken, moeten we eerst een andere query maken: een Union query. Kijk maar eens naar de brontabel die we gebruiken:

KindNR	Naam	Startdatum	Einddatum	Ma_VM	Di_VM	Wo_VM	Do_VM	Vr_VM	Ma_NM	Di_NM	Wo_NM	Do_NM	Vr_NM
1	Anton	1-3-2012	1-1-2015	☑	☑	☐	☐	☐	☑	☐	☐	☐	☑
2	Lisa	1-6-2012	1-5-2015	☑	☐	☑	☐	☐	☐	☑	☑	☑	☐
3	Jan	1-1-2012	30-6-2014	☐	☑	☑	☐	☐	☐	☐	☐	☐	☐
4	Piet	1-5-2012	31-3-2014	☑	☐	☑	☐	☑	☐	☑	☐	☑	☐
5	Fenna	1-1-2010	1-9-2012	☐	☑	☐	☐	☐	☑	☑	☐	☑	☑
6	Farah	1-3-2012	31-5-2012	☑	☐	☐	☑	☑	☐	☑	☐	☑	☐
7	Senna	1-2-2012	1-9-2012	☑	☑	☑	☐	☐	☐	☑	☐	☐	☑
8	Michaela	1-4-2012	1-12-2012	☑	☐	☐	☑	☑	☑	☑	☐	☑	☐
merking)				☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

Hierin zie je een schema opgenomen van dagdelen waarop de verschillende kinderen aanwezig zijn met de data waarop ze het dagverblijf bezoeken. De dagdelen waarop de kinderen aanwezig zijn worden vastgelegd met selectievakjes, waarvan de naam bestaat uit de verkorte dagnaam (Ma, Di etc) en de term VM voor de ochtend, en NM voor de middag. [Do_VM] houdt dus in: Donderdag middag.

Om het rapport te kunnen maken op basis van een maandselectie, moeten we eerst een genormaliseerde tabel maken van deze tabel. Daarmee bedoel ik een tabel waarin elk specifieke veld slechts eenmaal voorkomt. De naam komt bijvoorbeeld een keer voor, en dat is dus een genormaliseerd veld. Maar de entiteit *dagdelen* (bestaande uit een dagnaam en een aanduiding voor voormiddag of namiddag), die allemaal apart vermeld zijn, is dat niet. Elk dagdeel heeft nu namelijk een eigen veld. In een ideale situatie (genormaliseerde tabel) hebben we één veld voor de dag, en één veld voor het dagdeel. We moeten dus voor elk aangevinkt selectievakje een eigen record te maken waarbij de *naam* van het dagdeel in het veld Dagdeel wordt gezet, en de betreffende dag in het veld Dagnaam. Zo zullen er in de nieuwe query 4 records zitten voor Anton, en 6 records voor Michaela (kijk maar wanneer zij aanwezig zijn).

Voor deze query maken we een *Union* query. Ook dat querytype is al eerder behandeld, dus de uitleg hou ik kort: dit is een query waarbij je verschillende queries samenvoegt tot één query. Een Union query kun je niet in het Ontwerpvenster maken; hij moet in het SQL venster worden getypt. Gelukkig kun je met de basisquery (op basis van een gewone Selectiequery) beginnen, en daar later de overige elementen in plakken.

We gaan eerst een query maken waarbij voor het eerste dagdeel dat een kind aanwezig is een record wordt gemaakt. We beginnen met het filteren op het dagdeel VM, en op de dag Maandag. De basisquery voor deze query luidt:

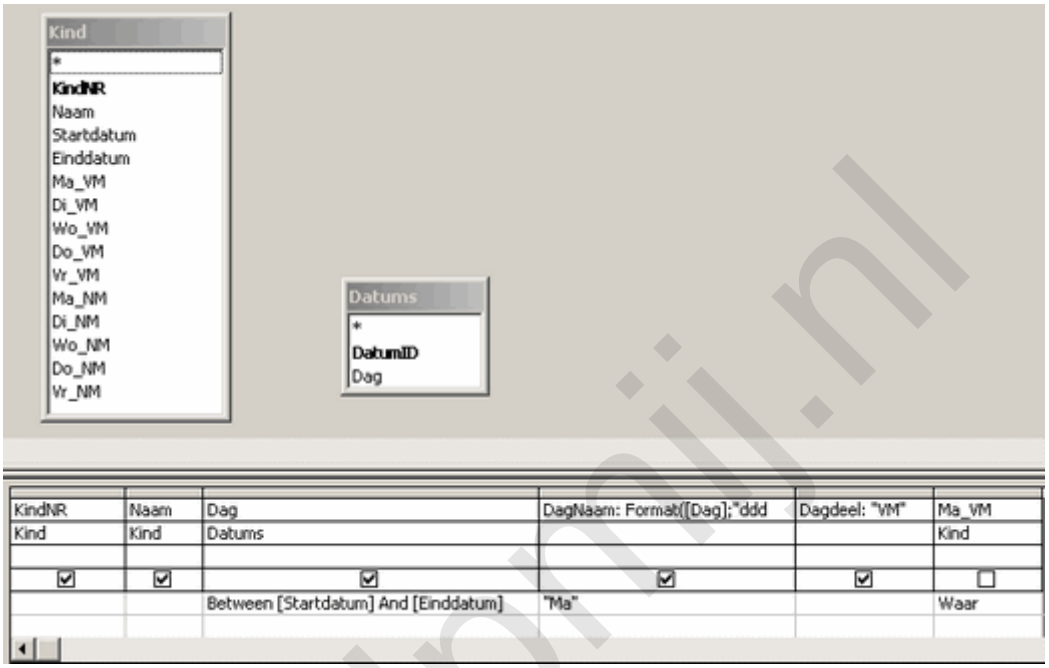
```
SELECT Kind.KindNR, Kind.Naam, Datums.Dag, Format([Dag],"ddd") AS DagNaam,
```

```
"VM" AS Dagdeel

FROM Kind, Datums

WHERE (((Datums.Dag) Between [Startdatum] And [Einddatum]) AND
((Format([Dag],"ddd"))="Ma") AND ((Kind.Ma_VM)=True));
```

En in het Queryvenster ziet dat er zo uit:



Wat misschien opvalt aan deze query, is het niet-gebonden veld [Dagdeel], dat we vullen met de tekst VM of NM, en niet met een waarde uit een veld. Het veld [Dagnaam] vullen we middels een formule met de naam van de dag.

Aan de regel FROM Kind, Datums kun je herkennen dat we een Cartesisch Product maken, omdat de twee tabellen niet aan elkaar gekoppeld zijn. Dat zie je ook terug in de afbeelding; er is geen verbindingslijn tussen de twee tabellen.

Samenvattend: de query gebruikt de velden Kind.KindNR, Kind.Naam uit de tabel [Kind], en het veld [Dag] uit de tabel [Datums]. Verder zie je dat er nog een verkorte dagnaam wordt gegenereerd op basis van het veld [Dag], en een nieuw veld wordt gemaakt met de naam [Dagdeel] en de waarde "VM".

De query wordt gefilterd op het veld [Ma_VM]. Omdat we alleen records willen zien in dit deel van kinderen die op Maandagochtend aanwezig zijn. Verder wordt het veld [DagNaam] gefilterd op de tekst "Ma" omdat we alleen de dag "Maandag" willen zien.

Deze query zou zonder verder criterium prima werken, en voor elk kind dat het veld Ma_VM aangevinkt heeft een record maken. Volgens de afbeelding zijn dat Anton, Lisa,Piet, Farah, Senna en Michaela.

Omdat een Cartesisch product alle datums laat zien die voldoen aan de criteria die we nu gebruiken, en de query dus geen rekening houdt met de datums [Startdatum] en [Einddatum], moeten we er nog een extra criterim aan toevoegen. Dat gebeurt op het veld [Dag] waar je een criterium ziet staan: Between [Startdatum] And [Einddatum]. En dit filter kijkt per datumrecord of de datum tussen de begin- en einddatum ligt, en die datums worden dan getoond. Kijken we bijvoorbeeld naar Farah, dan

krijgen we nu slechts deze records:

KindNR	Naam	Dag	DagNaam	Dagdeel
6	Farah	5-3-2012	Ma	VM
6	Farah	12-3-2012	Ma	VM
6	Farah	19-3-2012	Ma	VM
6	Farah	26-3-2012	Ma	VM
6	Farah	2-4-2012	Ma	VM
6	Farah	9-4-2012	Ma	VM
6	Farah	16-4-2012	Ma	VM
6	Farah	23-4-2012	Ma	VM
6	Farah	30-4-2012	Ma	VM
6	Farah	7-5-2012	Ma	VM
6	Farah	14-5-2012	Ma	VM
6	Farah	21-5-2012	ma	VM
6	Farah	28-5-2012	ma	VM

Farah is namelijk alleen aanwezig tussen 1-3-2012 en 31-5-2012.

Kortom: de query met het Cartesisch product doet precies wat we verwachten, en hiermee is het hoofdgedeelte van de query bepaald. Het enige dat we nu nog moeten doen, is de query uitbreiden, en er een UNION query van maken. Dat gaat het beste in een teksteditor zoals Notepad++, omdat je dan simpel kunt kopiëren en plakken. Om te beginnen moet je naar het SQL venster van de query, om de SQL code te kunnen bewerken. Daar zie je de code die hierboven ook staat terug. Deze code moet je selecteren, en kopiëren.

Vervolgens haal je de puntkomma aan het einde van de SQL weg, en typ je op een nieuwe regel: UNION.

Vervolgens maak je weer een nieuwe regel, en plak je de gekopieerde tekst. Omdat je nu twee keer dezelfde selectie maakt, wat een beetje onzinnig is, gaan we nu de tweede SELECT aanpassen aan de nieuwe voorwaarden.

- Verander de tekst *((Format([Dag],"ddd"))="Ma")* in **((Format([Dag],"ddd"))="Di")**
- Verander *((Kind.Ma_VM)=True))* in **((Kind.Di_VM)=True))**

Het resultaat ziet er dan zo uit:

```
SELECT Kind.KindNR, Kind.Naam, Datums.Dag, Format([Dag],"ddd") AS DagNaam,
"VM" AS Dagdeel

FROM Kind, Datums

WHERE (((Datums.Dag) Between [Startdatum] And [Einddatum]) AND
((Format([Dag],"ddd"))="Ma") AND ((Kind.Ma_VM)=True))

UNION

SELECT Kind.KindNR, Kind.Naam, Datums.Dag, Format([Dag],"ddd") AS DagNaam,
"VM" AS Dagdeel

FROM Kind, Datums

WHERE (((Datums.Dag) Between [Startdatum] And [Einddatum]) AND
```

```
((Format([Dag],"ddd"))="Di") AND ((Kind.Di_VM)=True));
```

Als je de query nu uitvoert, heb je de selectie uitgebreid; nu zie je de kinderen die op Maandagochtend en Dinsdagochtend aanwezig zijn. Door de SELECT tekst nog een te plakken, kun je de volgende selectie toevoegen.

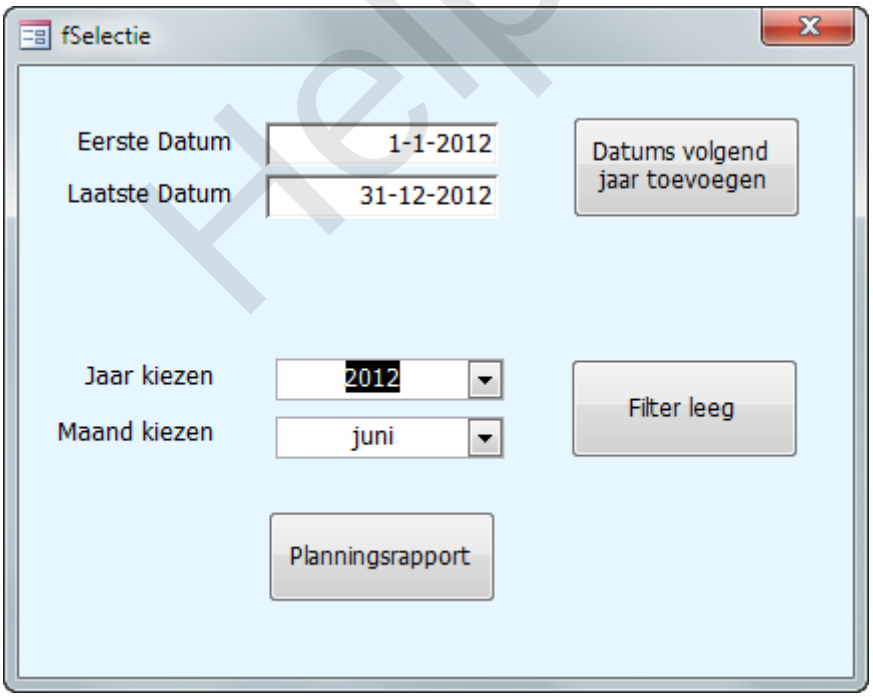
- Voeg onderaan de query het woord **UNION** toe op een nieuwe regel
- Plak de tekst nog een keer, en verander de volgende waarden:
- Verander *"VM" AS Dagdeel* in **"NM" AS Dagdeel**
- Verander *((Kind.Ma_VM)=True))* in **((Kind.Ma_NM)=True))**

Voer de query weer uit; je hebt nu Maandag Ochtend, Maandag Middag en Dinsdag Ochtend in de selectie zitten.

Je kunt nu zelf de overige dagdelen toevoegen. Als alles klaar is, kun je de query opslaan en uitvoeren. Je hebt nu een dynamische tabel (dat is immers een query) gemaakt die voor alle kinderen een record heeft gemaakt van de dagen en dagdelen waarop ze aanwezig zijn. Deze query zou zonder de tabel Datums dus niet te maken zijn! In deze cursus heb ik de query *qAanwezig* genoemd; ik zal die naam dan ook in het verdere verloop van dit hoofdstuk gebruiken. Deze query maakt een vrij stevige dynamische tabel aan; in mijn voorbeeld met 8 kinderen zijn dat er al zo'n 2900. Dat komt natuurlijk doordat er voor elk kind een groot aantal records nodig is om voor elke dag dat ze tussen de begindatum en de einddatum aanwezig zijn een of twee records te maken. En dat wordt alleen maar meer voor elk jaar waarvoor je records aanmaakt in tDatums!

Het keuzeformulier

In een eerdere afbeelding zag je een knop (en twee datumvelden) waarmee je de datums kon toevoegen aan de basistabel tDatums. Die afbeelding was een onderdeel van een groter formulier, dat je hier ziet.



In het formulier <fSelectie> zitten nog twee keuzelijsten waarmee je een jaartal kiest, en een keuzelijst om de maand voor het rapport te selecteren. Met de knop <Planningsrapport> worden de gegevens uit de query [qAanwezig] gefilterd, en toegewezen aan een tijdelijke query. Deze tijdelijke query wordt vervolgens aan het rapport gehangen. Het voordeel van een variabele query is dat de query altijd wel gegevens bevat, en het rapport dus ook. Je hoeft kortom niet eerst het formulier te openen om een

selectie te maken; je kunt het rapport zonder meer openen met de selectie die de laatste keer is gemaakt met behulp van het formulier.

Omdat we in dit formulier een aantal technieken gebruiken die al eerder zijn behandeld, volsta ik met het geven van de code. Die code is nu nog niet compleet, omdat de knop twee dingen doet: een nieuwe query maken (met de selectie), en het rapport openen dat is gebaseerd op de nieuwe selectiequery. Omdat we het rapport nog niet hebben, geef ik nu dus alleen het eerste deel, de code voor de query. Ik ga er zonder meer van uit dat je die zelf onder de knop kunt hangen!

Opdracht

- Maak een formulier met daarin twee keuzelijsten. In de eerste keuzelijst cboJaar moet je de jaartallen kunnen selecteren waarop kinderen zijn ingeschreven, en met de keuzelijst cboMaand de maanden die beschikbaar zijn voor dat betreffende jaar.

De selectie query

Zodra je op de knop <Planningsrapport> klikt, moet er een query gemaakt worden die de records uit de query [qAanwezig] filtert op de gekozen waarden. De techniek daarvoor is eerder gebruikt: we nemen een vaste query, en passen middels VBA code de SQL aan. Omdat duidelijkheid in naamgeving het werken wel zo plezierig maakt, krijgt de nieuwe query de naam [qAanwezig_Selectie]. De code kun je inmiddels zelf dus onder de knop hangen, en die ziet er zo uit:

```
Private Sub cmdPlanning_Click()

Dim dbs As DAO.Database

Dim qTmp As DAO.QueryDef

Dim strSQL As String, sFilter As String

On Error GoTo Err_cmdPlanning_Click



    strSQL = "SELECT Format([Datums]![Dag],'mmmm') AS Maand, Datums.Dag,
qAanwezig.Dagdeel, Count(qAanwezig.KindNR) AS AantalPers " _

        & "FROM Datums LEFT JOIN qAanwezig ON Datums.Dag = qAanwezig.Dag" &
vbCrLf

    strSQL = "SELECT Format([Datums]![Dag],'mmmm') AS Maand, Datums.Dag,
Datums.DagDeel, Count(qAanwezig.KindNR) AS AantalPers " _

        & "FROM Datums LEFT JOIN qAanwezig ON (Datums.DagDeel =
qAanwezig.Dagdeel) AND (Datums.Dag = qAanwezig.Dag)"

      If Not Me.cboJaar.Value = vbNullString Then

        sFilter = "WHERE (Year([Datums]![Dag]) = " & Me.cboJaar.Value & ")"

    End If

    If Not Me.cboMaand = vbNullString Then

        If Not sFilter & "" = "" Then sFilter = sFilter & " AND " Else:
```

```
sFilter = "WHERE "

        sFilter = sFilter & "(Month([Datums]![Dag]) = " & Me.cboMaand.Value
& ")"

    End If

    If Not sFilter & "" = "" Then strSQL = strSQL & sFilter & vbCrLf

    strSQL = strSQL & "GROUP BY Datums.Dag, Datums.DagDeel " & vbCrLf

    strSQL = strSQL & "ORDER BY Datums.Dag, Datums.DagDeel DESC;"



    Set dbs = CurrentDb

    Set qTmp = dbs.QueryDefs("qAanwezig_Selectie")

    qTmp.SQL = strSQL
```

Zoals je ziet, wordt de nieuwe SQL gemaakt op basis van de query [qAanwezig]. De functie is in essentie een Totalenquery; er wordt gegroepeerd op de velden [Dag] en [Dagdeel], en er is een veld [AantalPers] dat per dag en dagdeel het aantal personen telt. Het filter wordt apart opgebouwd op basis van de keuzelijsten op het formulier, en later samengevoegd met de SQL van de hoofdquery.

Enige uitleg, voor zover nodig bij de laatste 3 regels. Hier wordt de variabele *qTmp* (gedefinieerd als QueryDef) gebruikt om de (reeds bestaande; zou de query niet bestaan dan krijg je een foutmelding) query [qAanwezig_Selectie] opnieuw te definiëren.

Deze techniek is goed bruikbaar als je één query hebt voor een rapport, waar steeds een andere query(selectie) aan is toegewezen. Je zou deze techniek bijvoorbeeld kunnen overwegen als je een rapport hebt dat je wilt mailen met steeds een andere selectie. Bijvoorbeeld een factuur waarbij je een klant selecteert op een formulier. Het rapport baseer je dan op één query, en middels het aanpassen van de SQL van die query filter je steeds opnieuw op de juiste klant.

Het rapport Planningsrapport

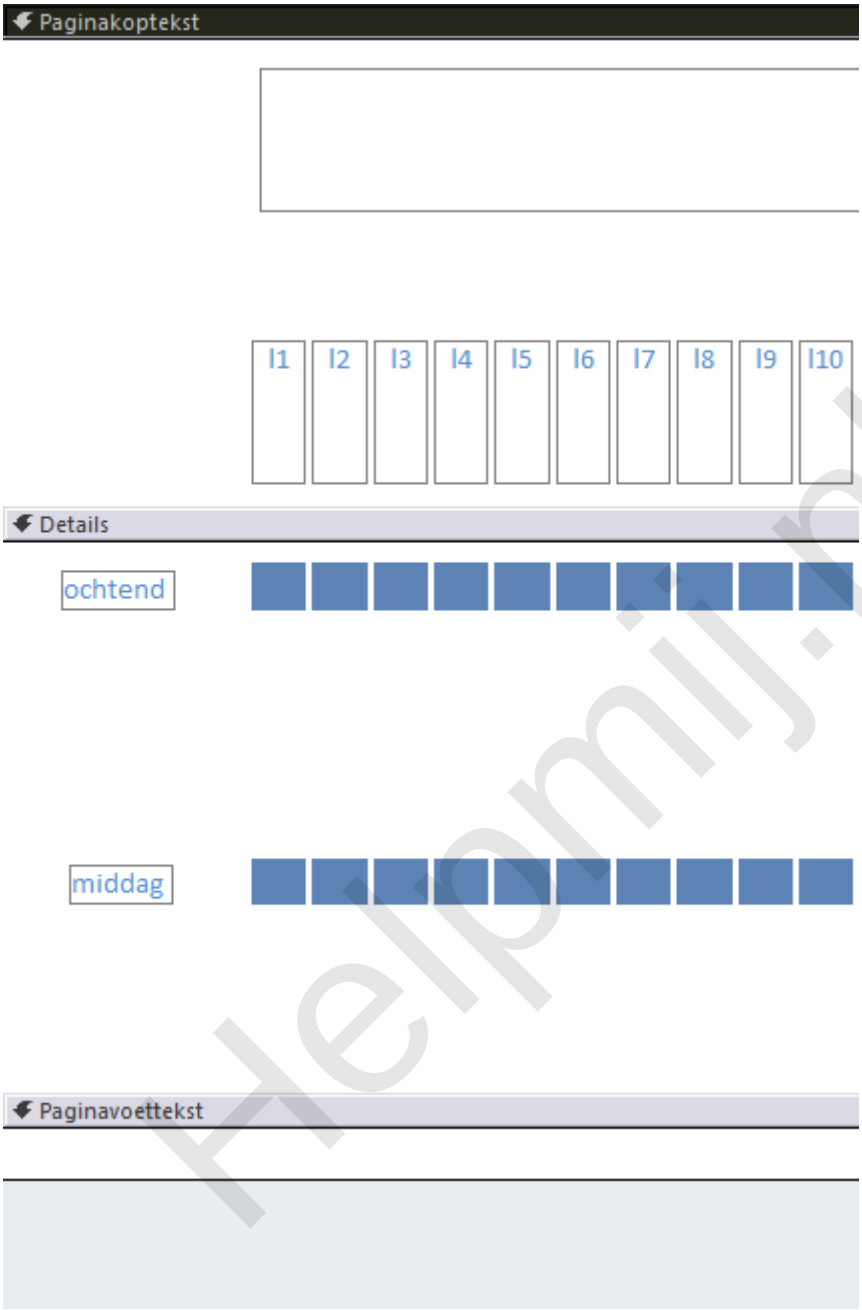
We gebruiken dus een niet-afhankelijk rapport voor de presentatie van het resultaat. We borduren hierbij voort op de techniek van het rapport dat we hebben gemaakt in de nieuwsbrief van April 2012, want ook het rapport uit dit hoofdstuk gaan we vullen a.d.h.v. een query bij openen (oplettende lezers hebben ook alang gezien dat ook het startformulier wel overeenkomsten heeft).

Het rapport dat we nu gaan maken, heeft velden die dynamisch gevuld gaan worden op basis van een query waarin dus al totalen zijn berekend, en een datumselectie is gemaakt (de query [qAanwezig_Selectie]. Enige probleem: omdat we twee dagdelen hebben, en voor elke dag dus twee records, valt er op het rapport nog niet zo eenvoudig een overzicht te maken. Het zou beter zijn als de data per dag in één record weergegeven kan worden. Gelukkig is dat vrij simpel, omdat we de data nu goed genormaliseerd hebben opgeslagen in de query [[qAanwezig_Selectie]. Je kunt namelijk op basis van deze query een Kruistabelquery maken, waarin je de velden [Maand] en [Dag] als rijkop gebruikt, het veld [Dagdeel] als Kolomkop en het veld [Aantalpers] als Waarde.

Het rapport wordt dus gebaseerd op deze Kruistabelquery, die de naam [qMaandbezetting_Selectie] krijgt.

De eerste stap in het ontwerp van het rapport bestaat uit het maken en plaatsen van de tekstvakken.

Een deel van het rapport zie je hieronder.



Je ziet hier de Paginakoptekst 10 van de in totaal 31 labels, en in de Details sectie 10 van de 31 tekstvakken voor het dagdeel Ochtend, en 10 van de 31 tekstvakken voor het dagdeel Middag. De reden dat er 31 tekstvakken zijn, is uiteraard dat er niet meer dan 31 dagen in een maand vallen.

In het voorbeeld dat in dit hoofdstuk gebruikt wordt, heb ik de labels benoemd van ‘l1’ t/m ‘l31’, de tekstvakken in de bovenste rij heten ‘v1v’ t/m ‘v31v’ en de tekstvakken in de onderste rij heten ‘v1n’ t/m ‘v31n’.

Doorgaans probeer ik in een rapport of formulier altijd namen te gebruiken die overeenkomen met de veldnamen in de onderliggende tabel of query, maar bij een niet-afhankelijk formulier kun je beter namen gebruiken die je in een lus kunt aanroepen. Dat gaan we namelijk doen bij het openen van het rapport.

De blauwe tekstvakken hebben allemaal dezelfde hoogte, en bevinden zich op dezelfde verticale

positie. Kijk je nog eens naar het voorbeeld aan het begin van het hoofdstuk, dan zie je dat de tekstvakken een grootte hebben die is aangepast aan het aantal personen voor dat dagdeel. Aan de hand van de waarden die we tegen gaan komen in de query (zie de query hieronder) bepalen we hoe hoog de hoogste kolom gaat worden. Deze waarde wordt dan gebruikt om de positie van de tekstvakken te bepalen.

Helaas gaat Access een beetje onhandig om met het positioneren van tekstvakken op een rapport: de positie wordt namelijk niet vanaf de onderkant van de sectie ingesteld, maar vanaf de bovenkant. Je kunt dus wel Object.Top instellen, maar niet Object.Bottom. En dat is dus wel eens lastig, want als je wilt weten op welke lijn 3 objecten van verschillende hoogte moeten staan, dan heb je niks aan het instellen van een vaste hoogte van die objecten. Dan staan ze netjes uitgelijnd aan de bovenkant, maar niet aan de onderkant!

Wil je de onderkant van een serie tekstvakken van ongelijke hoogte instellen, dan moet je dus de hoogte van het grootste object weten, en de (voor elk object gelijke) basis Topwaarde. Het berekenen van de positie is dus een klusje waar we speciale aandacht aan besteden!

De code begint weer met het declareren van de variabelen, en het uitlezen van de query en vullen van een matrixvariabele.

```
Private Sub Report_Open(Cancel As Integer)

Dim sVelden() As String, sMaand As String

Dim i As Integer, j As Integer, iMax As Integer, iHoog As Integer, iJaar As
Integer

Dim tmpHoog As Long, tmpTop As Long



    With CurrentDb.OpenRecordset("qMaandbezetting_Selectie")

        If .RecordCount > 0 Then

            .MoveLast

            .MoveFirst

            i = .RecordCount

            sMaand = .Fields("Maand").Value

            iJaar = Year(.Fields("Dag").Value)

            ReDim Preserve sVelden(i, 3)
```

In het eerste stuk wordt de query geopend, en wordt het aantal records opgezocht en toegewezen aan de variabele i. Het aantal dagen wordt vervolgens gebruikt om de dimensies van de matrix variabele vast te leggen. In de volgende fase wordt de matrix gevuld met de velden uit de query.

```
            Do While Not .EOF

                If .BOF Then
```

```
                    sMaand = .Fields("Maand").Value

                    iJaar = Year(.Fields("Dag").Value)

                End If                j = j + 1

                sVelden(j, 1) = .Fields("Dag").Value

                sVelden(j, 2) = .Fields("VM").Value

                sVelden(j, 3) = .Fields("NM").Value & ""

                If sVelden(j, 2) > iMax Then iMax = sVelden(j, 2)

                If sVelden(j, 3) > iMax Then iMax = sVelden(j, 3)

                .MoveNext

            Loop

            .Close

        End If

    End With

    On Error Resume Next

    iHoog = Me.v1v.Height * iMax
```

De matrix variabele is nu gevuld met de velden [Maand], [Dag], [VM] en [NM]. Je ziet ook dat er een waarde wordt toegekend aan de variabele iMax. Deze variabele wordt straks gebruikt om de positie te berekenen van de tekstvakken. De variabele moet het grootste getal krijgen, dus hij wordt alleen overschreven als één van de waardevelden een grotere waarde bevat dan op dat moment in iMax zit.

Je kunt overigens ook overwegen om twee iMax variabelen te declareren; één voor de bovenste lijn (VM), en één voor de onderste (NM).

De variabele iHoog wordt vervolgens bepaald a.d.h.v. de eerder ingestelde variabele iMax. Daarvoor wordt de hoogte van een willekeurig tekstvak genomen (ze zijn namelijk allemaal even hoog) en vermenigvuldigd met iMax. We weten nu dus de maximale hoogte van de kolommen.

Het hele proces wordt nu uitgevoerd in een lus.

```
    For i = LBound(sVelden) To UBound(sVelden)

        Me("l" & i).Caption = Format(sVelden(i, 1), "ddd") & vbCrLf &
Format(sVelden(i, 1), "dd")

        Me("v" & i & "v").Caption = sVelden(i, 2)
```

De lus wordt begonnen op basis van de laagste dimensiewaarde uit de matrix (Ubound(sVelden)), en loopt door tot de hoogste waarde (Ubound(sVelden)). In regel 2 wordt voor elke dag het *label* ingesteld; de naam van de dag en de kalenderwaarde. De derde regel zet de *waarde* van de dag in het label.

```
        tmpHoog = Me("v" & i & "v").Height * sVelden(i, 2)

        Me("v" & i & "v").Height = tmpHoog

        tmpTop = iHoog - tmpHoog

        Me("v" & i & "v").Top = (Me("v" & i & "v").Top + tmpTop)

        Me("v" & i & "v").TopMargin = (tmpHoog - 350)
```

Dit blokje bevat de berekening voor de hoogte, en dus ook voor de afstand vanaf de bovenrand. Nog even herhalen: je kunt van een object niet de plaats van de *onderkant* instellen, maar alleen de plaats t.o.v. de *bovenrand*. Om te weten hoe ver de onderkant van een object van de bovenrand af staat, moet je dus weten wat de hoogte is van het object. Die hoogte wordt bepaald door de standaardhoogte (die wordt uitgelezen met Me("v" & i & "v").Height)) te vermenigvuldigen met de waarde van het veld. De uitkomst wordt voor het gemak toegewezen aan de variabele tmpHoog. Bij de waarde 2 wordt het tekstobject dus 2 keer zo groot gemaakt, en bij 4 vergroten we het object 4 keer.

We hebben eerder al de maximale hoogte van het grootste object uitgerekend en aan de variabele iHoog toegekend. Door tmpHoog van iHoog af te trekken, weten we dus hoeveel pixels we het actieve object moeten verschuiven t.o.v. de bovenrand. Die uitkomst zetten we in de variabele tmpTop.

Nu kunnen we de definitieve positie instellen op het object. Die positie is namelijk de bestaande waarde (Me("v" & i & "v").Top) en daarbij tellen we dan tmpTop op.

```
        If sVelden(i, 2) > 0 Then Me("v" & i & "v").Visible = True Else:
Me("v" & i & "v").Visible = False
```

Simpele opdracht, na de vorige: als de waarde van het veld 0 is, dan moet het veld niet worden getoond. Je zou anders een dunne streep krijgen, en dat ziet er niet zo fraai uit.

In het volgende stuk code wordt het proces min of meer herhaald, maar nu voor de middag velden. Die staan al op een goede startpositie, dus er hoeft niet zoveel te worden veranderd behalve de objectnamen.

```
        Me("v" & i & "n").Caption = sVelden(i, 3)

        tmpHoog = Me("v" & i & "n").Height * sVelden(i, 3)

        Me("v" & i & "n").Height = tmpHoog

        tmpTop = iHoog - tmpHoog

        Me("v" & i & "n").Top = (Me("v" & i & "n").Top + tmpTop)

        Me("v" & i & "n").TopMargin = (tmpHoog - 350)

        If sVelden(i, 3) > 0 Then Me("v" & i & "n").Visible = True Else:
Me("v" & i & "n").Visible = False

        Me("l" & i).Visible = True

    Next i
```

De lus is klaar, en alle objecten geplaatst, hebben een waarde gekregen en zijn afhankelijk daarvan

zichtbaar gemaakt of niet. Als laatste worden de labels Ochtend en Middag uitgelijnd met de rij kolommen. En de titel wordt opgemaakt.

```
    Me.lblochtend.Top = Me.v1v.Top + Me.v1v.Height - Me.lblochtend.Height

    Me.lblmiddag.Top = Me.v1n.Top + Me.v1n.Height - Me.lblmiddag.Height

    Me.lblTitel.Caption = "Overzicht van " & StrConv(sMaand, vbProperCase)
& " " & iJaar

End Sub
```

Als laatste stap moet de code op het formulier nog worden uitgebreid. Die mist nog de code om het rapport te openen.

Opdracht

- Maak de code van de knop <cmdPlanning> zodanig af dat het rapport <Planningsrapport> wordt geopend.
- Test het rapport met verschillende selecties.

Samenvatting

In dit hoofdstuk hebben we een uitgebreid niet-afhankelijk rapport gemaakt dat voortborduurt op de techniek die we in april hebben behandeld. De opzet van het rapport is om een grafisch overzicht te maken met kolommen die de bezettingsgraad over een tijdperiode laat zien.

In het hoofdstuk zijn de volgende technieken behandeld:

- Een functie die een tabel vult met vaste waarden (datums in dit geval)
- De Union query (om gegevens van verschillende dagen samen te brengen)
- Een Kruistabel query (om de genormaliseerde gegevens te splitsen)
- Een procedure om een rapport te koppelen aan een wisselende gegevensbron
- Techniek om tekstvakken op een rapport te manipuleren (grootte, plaats)

De database die als basis is gebruikt voor dit hoofdstuk vind je in het Access forum, in de topic <Vragen, Opmerken, Reacties en Tips voor de Access cursus>. Hierin zit ook de uitwerking van de case.

Volgende aflevering

Omdat het de vakantieperiode is, en een redacteur ook wel eens aan iets anders denkt, volgende maand een aflevering met oplossingen voor veel voorkomende problemen. Als je het Access forum met enige regelmaat bezoekt, zal het je opvallen dat veel vragen vaak terugkomen. Voor een aantal daarvan zal ik een oplossingen aandragen, die algemeen toepasbaar zijn.