Cursus Access voor Beginners - Hoofdstuk 10 deel 1

Handleiding van Helpmij.nl

Auteur: OctaFish

November 2011

Formulieren: Een Stopwatch maken

Omdat het november is en het schaatsseizoen voor de deur staat, deze maand een aflevering die niet over duiken gaat, maar over schaatsen. Ouderen onder ons kennen vast de toernooischema's nog wel die bij elk groot toernooi in de krant verschenen. Een groot deel van het kijkplezier was toen het bijhouden van de uitslagen van de verschillende wedstrijden en het uitrekenen van de punten. Dat was toen ook een stuk makkelijker dan nu, omdat er behoorlijk wat langzamer werd geschaatst, dus er was veel meer tijd om de uitslagen en rondetijden op te schrijven!
Jammer genoeg is deze traditie geheel verdwenen; als je nu iets wilt bijhouden moet je zelf maar iets zien te brouwen in Excel, omdat je in Access weliswaar tijden kunt opslaan, maar de notatie ervan gaat niet specifieker dan Uren-Minuten-Seconden. En dat zou zelfs in de gouden tijden van Ard en Keessie niet voldoende nauwkeurig zijn geweest. Dus kunnen we in Access dan geen tijden vastleggen op duizendsten nauwkeurig? Wel, daar gaat deze aflevering dus over! Met een paar leuke trucjes kun je namelijk wel degelijk tijden registreren die op een duizendste nauwkeurig zijn.

We gaan in deze aflevering daarom een formulier maken waarbij we op basis van een via een keuzelijst te kiezen afstand twee rijders tegen elkaar kunnen laten rijden, en m.b.v. twee knoppen voor elke rijder de rondetijden kunnen vastleggen. Dat zal straks in de praktijk op de korte afstanden nog best lastig worden natuurlijk. Maar op de langere afstanden zal de nauwkeurigheid redelijk nauwkeurig kunnen zijn. Het formulier gaat er ongeveer zo uit zien:

Omdat er een hoop moet gebeuren voordat we het formulier als stopwatch kunnen gebruiken, en de ruimte in de nieuwsbrief uiteraard beperkt is, gaan we de opdracht in twee afleveringen uitvoeren. In de eerste aflevering leggen we uit de doeken hoe de timer zelf werkt. In de tweede aflevering passen we de code aan, zodat we de rondetijden voor de verschillende rijders kunnen vastleggen. En eventueel ook diskwalificaties en valpartijen, want die horen uiteraard ook bij een schaatswedstrijd!

Onderwerpen in deze aflevering

Waar gaan we in dit hoofdstuk mee aan de slag?

Functie GetTickCount

Om met nauwkeurige tijden te kunnen rekenen, zullen we gebruik moeten maken van een standaard functie van Windows zelf: de functie GetTickCount. Wat deze functie doet, is vrij letterlijk het aantal interne 'tikken' tellen op basis van milliseconden. Deze telling gebeurt op basis van de tijd waarop de computer is opgestart, dus zodra je de pc aanzet, begint de teller te tellen. Als je dus twee metingen neemt met een bepaald interval, dan heb je de verstreken tijd in milliseconden. Het enige dat we dan nog moeten doen, is de uitkomst omzetten naar een voor ons leesbaar formaat, voor ons formulier is dat: mm:ss,00. Minuten, seconden en duizensten van seconden dus. Een formulier heeft een eigenschap <Timer> die we gaan gebruiken om de tijd te starten of te stoppen, zodat we de begin- en

eindtijd kunnen vastleggen en berekenen. En daarmee de rittijd kunnen uitrekenen. Voor de tussentijden pakken we een momentopname, namelijk het klikken op een knop.

Onafhankelijke recordsets

Eén van de problemen waar we tegenaan gaan lopen, is dat alle handelingen die je op een formulier (of in een database in het algemeen) uitvoert, tijd kosten. En die bewerkingstijd gaat de nauwkeurigheid van de tijdregistratie behoorlijk in de weg zitten. Om de processor zoveel mogelijk te ontzien, is het daarom belangrijk om de computer zo min mogelijk te laten rekenen of uitvoeren. Daarom maken we op het formulier gebruik van een *onafhankelijke recordset*. Dat is een virtuele tabel die precies zo werkt als een gewone tabel, maar die alleen in het geheugen van de computer bestaat. Een dergelijke tabel is prima geschikt om tijdens de race de tussentijden op te slaan. Als beide rijders over de finish zijn, is er tijd genoeg om de bewaarde gegevens op te slaan door de tabel uit te lezen, en op te slaan in een normale tabel.

Matrix variabelen

Behalve een onafhankelijke recordset kunnen we de tussentijden ook opslaan in een matrixvariabele. Ook deze techniek komt daarom aan bod, want matrixvariabelen zijn zeer geschikt om veel bij elkaar horende gegevens op te slaan. Het onderwerp is dus ook prima geschikt om daar aandacht aan te besteden. Bij een matrixvariabele leg je eerst de omvang van de matrix vast (gegevenstype, dimensionaliteit) waarna je de matrix kunt vullen. Er is voor ons formulier wel een verschil in werking, en die zal straks bepalen welke methode je zult willen gebruiken.

Verschillen tussen een onafhankelijke recordset en een matrix variabele

Het grote verschil is, om te beginnen: een recordset is een virtuele tabel, die je helemaal zelf kunt inrichten. Net als in een gewone tabel geef je aan wat voor soort velden je wilt gebruiken, zoals tekstvelden of numerieke velden. Een recordset is dus heel flexibel. In tegenstelling tot een matrix variabele, waarbij je eerst vast moet leggen uit hoeveel dimensies hij moet bestaan, en van wel type de variabele moet zijn. Daarbij kun je kiezen voor de bekende gegevenstypes, waarbij het type Variant de meeste geheugenruimte opeist, maar wel alle soorten gegevens kan opslaan.

Een tweede verschil is, dat een recordset met een simpel commando is te sorteren, net zoals je een tabel sorteert. Een matrixvariabele is alleen met een ‘brute kracht’ te sorteren. Je moet daarbij alle waarden van de matrix uitlezen en vergelijken met de vorige om te zien of hij eerder of later moet komen. Daarom is de matrix variabele beter geschikt voor gegevens die je niet bij voorbaat gesorteerd hoeft te hebben. Wil je gegevens wel sorteren voordat je ze opslaat, dan is de onafhankelijke recordset een betere keus.

De inrichting van de database

Omdat we in eerdere hoofdstukken al uitgebreid hebben stilgestaan bij de inrichting van een database, zal ik de inrichting hier niet uitgebreid belichten, maar alleen de hoofdpunten van de database aanstippen, voor zover ze van invloed zijn op het formulier.

Om te beginnen, hebben we een tabel [tRitten] nodig waarin we de rituitslagen gaan vastleggen. Deze tabel bevat in ieder geval een ID-veld, de ID’s van de twee rijders en de te rijden afstand. De tabel heeft een één-op-veel relatie met de tabel [tRittijden], waarin we de RitID als koppelveld terugzien, een veld SchaatserID (dat kan uiteraard slechts één van de twee schaatsers uit het bijbehorende Rit record zijn), een veld voor de verreden afstand, en de geregistreerde tijd.

Het formulier bestaat uit een Hoofdformulier met de ritgegevens, en een subformulier met de bij de rit behorende tussentijden. Deze tabel (en dus het subformulier) wordt gevuld als de rit is afgelopen. Zoals hierboven al uitgelegd, is het belangrijk voor een goede tijdwaarneming dat de computer zo weinig mogelijk hoeft te doen, zodat de tickercount functie optimaal kan werken.

Om de tijden zo nauwkeurig mogelijk vast te leggen, willen uiteraard de snelste manier gebruiken om een tussentijd of eindtijd vast te leggen. Dat de muis daar totaal ongeschikt voor is hoeft hoop ik geen betoog; daarom bouwen we het formulier op bediening met het toetsenbord. Idealiter hebben we op het formulier dan ook één of twee knoppen, die om en om actief zijn. Is er in een rit maar één rijder actief, dan heb je uiteraard aan een knop genoeg. Zijn er twee rijders op de baan, dan wil je elke rijder kunnen vastleggen zodra hij over de streep is. Dat betekent, dat na een eerste druk op de knop, als je de eerste tussentijd vastlegt, de andere knop actief wordt, zodat je met een druk op <Enter> de volgende tijd kunt vastleggen.

Omdat je niet van te voren kunt vastleggen welke rijder de eerste tijd zal maken, zul je zelf op moeten letten welke knop je als eerste moet indrukken. Daarna zou het systeem automatisch de volgende knop moeten activeren, zodat je bij het passeren van de finishlijn alleen maar op <Enter> hoeft te drukken om de volgende tijd te registreren. Je zou die actie misschien ook nog aan de spatiebalk kunnen toewijzen, om nog makkelijker te kunnen vastleggen.

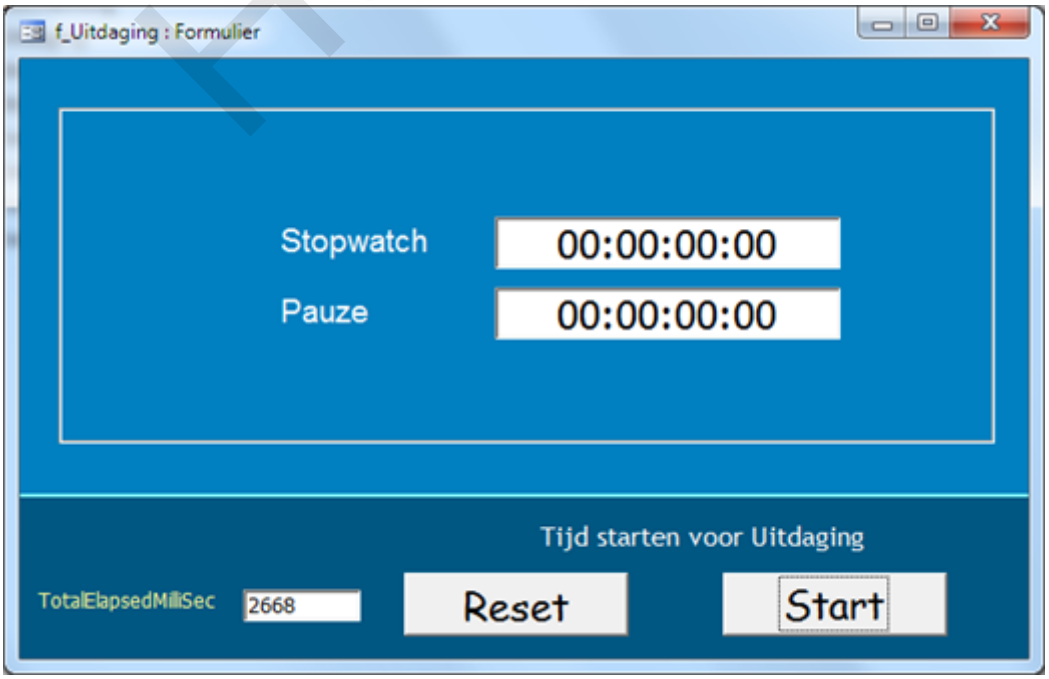
Bij schaatswedstrijden zijn gelukkig een aantal gegevens eenvoudig te bepalen. Zo bestaat elke rit uit een vooraf bepaalde afstand, en heeft de baan een vaste lengte; elk rondje dat een rijder moet rijden is 400 meter lang. Jammer genoeg zijn de te rijden afstanden daar niet op uitgekozen; de schaatswedstrijden bestaan (bij de heren, om het niet te ingewikkeld te maken) uit ritten van 500 meter, 1500 meter, 5000 meter en 10.000 meter. Alleen de laatste afstand is een veelvoud van 400 meter, namelijk 25 rondjes. Bij de overige afstanden wordt daarom eerst een tijdmeting gedaan na een kortere afstand. Op het formulier moeten we daar rekening mee houden. We bepalen dus, als we de tussentijden gaan vastleggen, ook de afstand die bij de eerste meting is gereden. We doen dat door de restafstand te berekenen met de functie Mod.

Verder willen we uiteraard dat de tijdmeting stopt als de schaatser de volledige afstand heeft gereden. Dat doen we door de knop waarmee we de tussentijden vastleggen uit te schakelen als de volledige afstand is gereden.

Alles bij elkaar genoeg technieken om een regenachtige zondagmiddag te vullen!

De Timer functie

Bij een stopwatch gaat het uiteraard in eerste instantie om het berekenen van de verstreken tijd. Daarom bekijken we eerst een simpele opzet van het formulier.



De stopwatch wordt gestart met een Start/Stop knop. De werking daarvan spreekt hopelijk voor zichzelf.

Als we het formulier in de ontwerpfase openen, zien we een aantal gebeurtenissen: we zien een procedure bij de gebeurtenis <Bij Aanwijzen>, en een procedure bij de gebeurtenis <Bij timer>. Laten we de VBA code eens bestuderen. Druk dus op <Alt>+<F11> om naar de code te gaan, of gebruik de knop met de drie puntjes om naar één van de procedures te gaan. Ga vervolgens helemaal naar boven, naar de algemene sectie van de code.

Declareren van de algemene variabelen

We hebben een aantal variabelen nodig om de tijdelijke gegevens in op te slaan. Die moeten eerst worden gedeclareerd. Overigens is het declareren van variabelen niet nodig als je de regel *Option Explicit* weg laat, maar dat is niet aan te bevelen. Door je variabelen op de juiste manier te declareren, heb je de beste controle over de werking van je code.

```
Option Compare Database

Option Explicit

Private Declare Function GetTickCount Lib "kernel32" () As Long

Dim TotalElapsedMilliSec As Long

Dim StartTickCount As Long

Dim bTijdLoop As Boolean
```

De eerste twee regels zijn ongeveer standaard voor Access. De derde regel is echter essentieel voor de counter functie. Hierin wordt de functie GetTickCount gedeclareerd. Omdat Access zelf geen counter functie kent, maar Windows wel, wordt een verwijzing gemaakt naar een windows object: Lib "kernel32" . Hier is verder niet zoveel over te zeggen, al is de kernel voor programmeurs wel belangrijk. In het Kader van een cursus Access heeft het niet zoveel zin om er dieper op in te gaan. Het volstaat hier om te weten hoe we de functie ‘er uit kunnen lichten’.

Er worden nog wat algemene variabelen gedeclareerd, net als de functie GetTickCount van het numerieke type Long, en dat is omdat de waarden meestal te groot zullen zijn voor kleinere variabelen als Integer. De variabelen worden in het algemene gedeelte gedeclareerd, zodat we ze op verschillende momenten kunnen aanroepen en vullen. Zoals je misschien weet, hebben variabelen een bepaalde levensduur. Als je ze binnen een subroutine van een knop declareert, houden ze alleen hun waarde binnen die routine. Bij een klok of een stopwatch is dat niet handig, want daar moeten we de waarden vasthouden gedurende de hele tijdsmeting. Door een variabele niet in een procedure te declareren, maar in het algemene gedeelte, dus bovenin de module, kun je een variabele vullen, en vanuit een andere procedure opvragen en verder bewerken.

De gebeurtenis <Bij aanwijzen>

Op het formulier moeten standaardwaarden worden ingesteld. Dat doen we bij de gebeurtenis <Bij aanwijzen>, al zou dat in dit geval ook kunnen bij de gebeurtenis <Bij Laden>, omdat het instellen maar één keer hoeft te gebeuren. Het formulier is namelijk niet gebonden aan een gegevensbron, dus we kunnen niet bladeren door eventuele records.

```
Private Sub Form_Current()

    Me.ElapsedTime = "00:00:00,000"
```

```
    Me.PauseTime = "00:00:00,000"

    TotalElapsedMilliSec = 0

    Me.TimerInterval = 0

End Sub
```

Het formulier kent twee procedures: de gebeurtenis <Bij Aanwijzen> bevat bovenstaande code. De tekstvakken ElapsedTime en PauseTime krijgen bij het laden van het formulier een standaardwaarde van "00:00:00:00". Zoals je ziet, is dat een opmaak die overeenkomt met Uren:Minuten:Seconden,Duizendsten. Een oplettende lezer zal het misschien zijn opgevallen dat er geen Format opdracht wordt gebruikt; iets dat toch gebruikelijk is bij de opmaak van tijden. De reden daarvan komt later aan bod.

Na het instellen van de tekstvakken, wordt de teller **TotalElapsedMilliSec** op de waarde 0 ingesteld. De laatste regel zet de formuliereigenschap TimerInterval op 0. In essentie wordt hiermee de timerfunctie van het formulier uitgezet.

De gebeurtenis <Bij aanwijzen>

Het echte werk gebeurt op de gebeurtenis <Bij Timer> van het formulier. Laten we deze code eens bekijken:

```
Private Sub Form_Timer()

Dim Hours As String, Minutes As String, Seconds As String, MilliSec As String
Dim ElapsedMilliSec As Long

ElapsedMilliSec = (GetTickCount() - StartTickCount) + TotalElapsedMilliSec

Hours = Format((ElapsedMilliSec 3600000), "00")

Minutes = Format((ElapsedMilliSec 60000) Mod 60, "00")

Seconds = Format((ElapsedMilliSec 1000) Mod 60, "00")

MilliSec = Format((ElapsedMilliSec Mod 1000), "000")

If Me.btnStartStop.Caption = "Stop" Then

    Me.lblKlok.Caption = Format(Now(), "HH:mm:ss")

    Me.ElapsedTime = Hours & ":" & Minutes & ":" & Seconds & ":" & MilliSec

End If

End Sub
```

De Timer procedure gebruikt een aantal variabelen, die alleen binnen de Timer functie gebruikt worden. Deze worden dus binnen de functie gedeclareerd, en niet algemeen. De variabelen Hours, Minutes, Seconds en Millisec worden allemaal opgemaakt als *String*. Dat lijkt onlogisch, omdat ze alleen getallen zullen bevatten. Je zou ze daarom ook als getal kunnen declareren. We doen dat hier niet, omdat het nu iets eenvoudiger is om de berekende uren, minuten etc. op te maken met een voorloopnul. En, misschien wel de belangrijkste reden, het eindresultaat, de verstreken tijd, is in een opmaak die Access niet kent. En dus ook niet is op te maken in een standaard Access opmaak. Het

eindresultaat is dus een variabele die we toch al als tekst op moeten maken. En dat verklaart dus ook de opmaak die we in de gebeurtenis <Bij aanwijzen> zagen: de tekstvakken krijgen straks tekst toegewezen, en geen getallen of datums.

De verstreken tijd wordt verrekend met deze formule:

ElapsedMilliSec = (GetTickCount() - StartTickCount) + TotalElapsedMilliSec

Hierbij wordt de gemeten GetTickCount waarde gepakt, daar wordt de startwaarde van afgetrokken, en de totaaltijd weer bij opgeteld. En die functie GetTickCount is dus het hart van de berekening. Wat deze functie doet, is het volgende:

Berekent het aantal millisecondes dat is verstreken sinds het systeem is gestart.

Je zult begrijpen dat het lezen van millilsecondes niet al te eenvoudig is; maar gelukkig kun je met een paar simpele berekeningen uit het totaal aantal millisecondes wel de uren, minuten en secondes berekenen. Dat gaat, zonder de opmaak uit de code hierboven, als volgt:

Hours = (ElapsedMilliSec 3600000)

Minutes = (ElapsedMilliSec 60000) Mod 60

Seconds = (ElapsedMilliSec 1000) Mod 60

MilliSec = (ElapsedMilliSec Mod 1000)

Het is niet zo moeilijk om te zien hoe deze berekeningen zijn opgebouwd. Neem de berekeningen voor uren: als je 3600000 eerst door 60 deelt om de minuten te vinden (resultaat: 60000), en daarna nog een keer door 60 (voor de seconden) houd je het getal 1000 over; precies één seconde dus. Kortom: 1000 milliseconden is exact één seconde. En dat volgt ook eigenlijk wel uit de naam J.

Om de minuten en seconden te vinden, gebruiken we de functie MOD 60. Daarbij wordt de waarde ElapsedMilliSec door 60 gedeeld, en is de uitkomst de waarde die over blijft. De restwaarde dus. Om het aantal Millisecs te berekenen, delen we de verstreken tijd dus door 1000, om het gehele getal te krijgen. Omdat alle getalvariabelen gedeclareerd zijn als Long, hoeven we geen rekening te houden met afrondingen; alle getallen zijn gehele getallen.

De gebeurtenis <Bij Klikken> van de knop

We hebben de berekening klaargezet, en het formulier ingesteld op de startwaarden. Het enige dat nog rest, is om de teller te starten en te stoppen. Dat doen we met de code die we onder een knop zetten. De Start/Stop knop heeft de volgende code:

```
Private Sub btnStartStop_Click()

If Me.TimerInterval = 0 Then

    StartTickCount = GetTickCount()

    Me.TimerInterval = 10

    Me!btnStartStop.Caption = "Stop"

    Me!btnReset.Enabled = False

Else
```

```
    TotalElapsedMilliSec = TotalElapsedMilliSec + (GetTickCount() - StartTickCount)

    Me.TimerInterval = 0

    Me!btnStartStop.Caption = "Start"

    Me!btnReset.Enabled = True

End If

End Sub
```

We gebruiken één knop om te starten en te stoppen. Een groot voordeel daarvan is, dat je je niet kunt vergissen in de werking van de knop. En we kunnen acties triggeren op basis van een vast gegeven, bijvoorbeeld de tekst op de knop, de *Caption*.

In dit geval schakelen we op basis van de Timer instelling van het formulier. We willen de tijd kunnen starten en stoppen. Een formulier waarbij de TimerInterval is ingesteld op 0, staat, als het om tijdwaarneming gaat, stil. Elke waarde hoger dan 0 betekent dat het formulier gaat herberekenen en de timer dus werkt. De minimale waarde om de herberekening in te stellen is derhalve 1: hierbij wordt de actie GetTickCount elke milliseconde uitgevoerd. Omdat weinig mensen dat nog onderscheidend kunnen waarnemen, is dat niet zo'n handige waarde. Met *Me.TimerInterval = 10* wordt de interval ingesteld op 1/100 seconde. Je kunt ook *Me.TimerInterval = 100* gebruiken, om het formulier elke 1/10 seconde bij te werken.

Een klok op je formulier

Een waarde die je vaak tegenkomt, is *Me.TimerInterval = 1000*. Hierbij wordt het formulier dus exact elke seconde ververst. Als je een klok wilt maken op een formulier, die elke seconde wordt bijgewerkt, dan is 1000 een prima waarde. Een manier om een 'levende' klok op je formulier te zetten is de volgende. Je gebruikt daarvoor weer de gebeurtenissen <Bij Aanwijzen> en <Bij Timer>. Om het voorbeeld te gebruiken, moet je eerst een label maken met de naam lblKlok. Een andere naam mag uiteraard ook, maar dan moet je de code aanpassen.

```
Private Sub Form_Current()

    Me.lblKlok.Caption = Format(Now(), "HH:mm:ss")

    Me.TimerInterval = 1000

End Sub
```

Deze code geeft het bijschrift van het label *lblKlok* de huidige tijd. Om het formulier, en dus het label, elke seconde bij te werken, zetten we deze code ook bij de gebeurtenis <Bij Timer>.

```
Private Sub Form_Timer()

    Me.lblKlok.Caption = Format(Now(), "HH:mm:ss")

End Sub
```

De crux van de Start/Stop knop

Maar we zijn nog niet klaar met de start/stop knop… Eerst nog even de code erbij pakken.

```
If Me.TimerInterval = 0 Then

   Me.TimerInterval = 10

   Me.btnStartStop.Caption = "Stop"

   StartTickCount = GetTickCount()

Else

   Me.TimerInterval = 0

   Me.btnStartStop.Caption = "Start"

   TotalElapsedMilliSec = TotalElapsedMilliSec + (GetTickCount() - StartTickCount)

End If
```

We hebben al gezien dat om de tijdwaarneming te stoppen, we de TimerInterval de waarde 0 moeten geven. Om de code weer op te starten, moeten we daarna de waarde instellen op het door ons gewenste interval. De hele actie kunnen we dus triggeren op de feitelijke waarde van het timerinterval. Vandaar dat we het starten en stoppen bepalen met een IF vergelijking op de ingestelde intervalwaarde. Met If Me.TimerInterval = 0 Then kijken we eerst naar de huidige waarde van het interval. Als die waarde 0 is, staat de klok uit. Dat betekent, dat de teller daarna moet worden gestart, en dat er een waarde moet worden ingesteld. Dat houdt tevens in, dat de tekst op de knop nu “Start” luidt, en dat we die moeten veranderen in “Stop”. De om het tijdverschil te kunnen berekenen hebben we een startwaarde nodig. Die halen we op met GetTickCount (dezelfde functie die we ook gebruiken om het verschil te meten) en slaan we op in de variabele *StartTickCount*.

De tweede tak van de IF bevat de code die wordt uitgevoerd als de timerinterval de waarde 10 heeft. Al zou ik uiteraard moeten zeggen: niet gelijk is aan 0. Dan gebeurt min of meer het omgekeerde: de timer wordt stilgezet door hem de waarde 0 te geven, en het bijschrift op de knop wordt veranderd in “Start”. De totaaltijd wordt nu ook berekend op basis van GetTickCount en StartTickCount.

Samenvatting

Access kan prima met datums en kloktijden overweg, maar kan niet rekenen in honderdsten of duizendsten van seconden. Wil je dat wel kunnen, dan moet je een Windows functie aanroepen, namelijk de functie GetTickCount. Deze functie telt de verstreken tijd sinds het aanzetten van de computer in milliseconden. Deze functie heeft wel een beperking: hij kan niet verder tellen dan 49,7 dagen. Dus voor die tijd moet je de pc een keer herstarten…

Een formulier heeft een Timer eigenschap, waarmee je een interval kunt instellen waarop een formulier acties moet gaan uitvoeren. Deze eigenschap gebruiken we in het voorbeeld om een klok te maken, of verstreken tijd te berekenen. Daarbij leggen we eerst met GetTickCount een startwaarde vast, en lezen na de verstreken tijd een eindwaarde vast. Het verschil tussen de twee is de verstreken tijd in milliseconden. Door de juiste berekeningen te maken, kunnen we uit deze verstreken tijdswaarde een tijd terugrekenen. Omdat Access tijd alleen in seconden als laagste differentiatie kan berekenen, moeten we van de tijd een eigen opmaak maken. De tijd wordt dan ook een tekstveld, dat er uit ziet als een tijdblok met duizendsten of honderdsten seconden.

Volgende Aflevering

In het volgende hoofdstuk gebruiken we de technieken van de stopwatch om tijdblokken vast te leggen in een recordset, en in een matrix variabele. Om, als een race gefinisht is, de opgeslagen tijden weg te schrijven in een tabel.