Cursus Access voor beginners

Handleiding van Helpmij.nl

Auteur: Octafish

Januari 2011

Inhoudsopgave

Cursus Access voor Beginners

Iedereen die wel eens op het internet heeft gezocht naar een Access cursus, kent ze wel: de cursussen waarbij naar een paar inleidende woordjes al gelijk de eerste knoppen ingedrukt worden: we gaan een database maken! Waarom zou Helpmij.nl daar nog een cursus aan toevoegen? Antwoord op deze vraag: omdat een database maken nou juist *niet* begint met een druk op de knop *Nieuw...*

Introductie

In tegenstelling tot de meeste applicaties in de Microsoft Office, begint het ontwerpen van een database op papier, en in het hoofd: met een *denkproces*. Als je een pakket als Word, Excel of PowerPoint opstart, zie je een leeg beginscherm, waarin je gelijk kunt gaan werken. In Word begin je met typen, en als je uitgetypt bent, is je document eigenlijk al klaar. Nog wat opmaak veranderen, en je kunt printen of mailen. Hetzelfde geldt in min of meerdere mate voor Excel en PowerPoint. Het is, zoals Microsoft het ook bedoeld heeft; WYSISYG (What You See Is What You Get). Opslaan, sluiten en op naar het volgende document! Daarbij geldt niet alleen WYSISYG, maar vaak ook YOYO (You're On Your Own). Je maakt je eigen documenten, beheert ze vaak zelf, en verspreidt ze zelf. Zo af en toe werk je met andere mensen aan een document, waarbij je dan wijzigingen kunt redigeren, maar veel verder gaat het samenwerken vaak niet.

Bij Access werkt het allemaal iets anders. Meestal maak je voor jezelf of je werk maar een paar databases, waar vervolgens andere mensen in gaan werken. Daarbij is het niet de bedoeling dat anderen iets *veranderen* aan de database, maar wel gegevens *muteren* en *toevoegen*. De databasebeheerder regelt verder het beheer van de database. Dat betekent, dat voordat je anderen kunt laten werken in je database, deze al helemaal af moet zijn: alle tabellen moeten de juiste gegevens kunnen bevatten, alle queries moeten de juiste gegevens en management info kunnen leveren, alle formulieren moeten werken en voor de gebruikers duidelijk in het gebruik zijn, en alle rapporten moeten de juiste output opleveren. En dat betekent weer, dat je voordat je gaat bouwen, eerst moet inventariseren wat de wensen van de verschillende (groepen) gebruikers zijn. Wat moet er worden opgeslagen, en vooral: wat moet er uit kunnen worden gehaald?

Microsoft Access is in dit geheel dus een buitenbeentje. Het is ook niet voor niks dat Access geen onderdeel uitmaakt van een standaard Office suite. Pas vanaf de versie Professional is Access als onderdeel toegevoegd, en passant de prijs van de suite zo ongeveer verdubbelend…
Een database maken is dus geen sinecure. Het vereist planning, en een zorgvuldige uitvoering. Hoeveel gebruikers gaan de database gebruiken? Moeten er verschillende omgevingen worden gemaakt voor verschillende gebruikersgroepen? Etc…

Opzet van deze cursus

In deze cursus gaan we een volledig database systeem maken. De database maakt gebruik van een *Frontend* database, en een *Backend* database. De *connectie* tussen de frontend en de backend wordt via VBA protocollen gelegd, waarbij de gebruikers zo kort mogelijk verbonden zijn met de database. Dit geeft de gebruikers een systeem waarbij ze op een optimale manier gebruik kunnen maken van de database, en de database zo min mogelijk wordt belast, waardoor er relatief veel personen tegelijk met de database kunnen werken.

In de eerste les van de cursus gaan we in op het *opzetten* van de database. In de eerste fase hebben we nog geen database pakket nodig; pen en papier is bij wijze van spreke al genoeg. We gaan inventariseren welke activiteiten we in de database willen kunnen uitvoeren. Dit model wordt later vertaald naar de feitelijke database.

De tweede fase bestaat uit het *Modelleren* van het systeem. Hierbij wordt schematisch vastgelegd welke activiteiten er in de database moeten worden ondergebracht. Als er bijvoorbeeld alleen contante winkelactiviteiten worden uitgevoerd, dan zal het niet nodig zijn om een module Facturen te maken, omdat er immers geen facturen worden geleverd. Als na verloop van tijd blijkt dat het toch nodig is om de

transacties vast te leggen, moet het systeem worden aangepast, omdat het factureren in eerste instantie niet is gebouwd. Door van te voren na te gaan wat we van het systeem verwachten, kunnen we deze problemen voor zijn.

Het modelleren van een database gebeurt in een ER-diagram. Dit staat voor: Entiteit/Relaties gegevensmodel. Daarbij is een *Entiteit* een object waarover informatie moet worden bijgehouden, en een *Relatie* een verbinding tussen twee entiteiten. Deze begrippen worden verder nog uitgebreid behandeld.

In de derde fase gaan we bekijken welke *gegevens* we gaan opslaan en gebruiken. Hierbij kijken we naar welke gegevens we in welke tabellen gaan gebruiken. De vraag die daarbij aan bod komt is: hoe kunnen we voorkomen dat dezelfde gegevens in verschillende tabellen opnieuw moeten worden ingevoerd? Het meermalen opslaan van dezelfde gegevens heet in *Redundantie*[i]. Bij het ontwerpen van een database streven we ernaar om gegevensredundantie zoveel mogelijk te voorkomen. Dat heeft verschillende voordelen:

- Wijzigingen hoeven vaak maar in één tabel te worden doorgevoerd
- Er is geen gevaar voor inconsistente informatie
    - De tabellen worden niet nodeloos groot

We noemen dit het *Normaliseren* van de database. Hiervoor zijn een aantal regels opgesteld, waaraan we de tabellen kunnen toetsen. Uiteindelijk leidt dit tot een tabellenstructuur voor de database.

In de vierde fase kunnen we de tabellen gaan maken in een database pakket. In deze cursus gebruiken we daarvoor als basis Access 2003, maar alle oefeningen zijn ook te maken in de nieuwere versies van Access. Daar waar de schermen/handelingen afwijken, zal dat worden aangegeven.

De case voor de cursus

We gaan in deze cursus een database maken voor een Multi-purpose gebruik: een nieuw op te starten Duikcentrum. Dit duikcentrum beheert onder andere een verkoopwinkel, organiseert trainingen en verhuurt duikspullen. Het duikcentrum bestaat daarmee uit drie hoofdtakken:

1. Een duikclub
2. Een duikwinkel
3. Een verhuurcentrum

De Duikclub

De duikclub heeft als belangrijkste taak het organiseren van duikevenementen en trainingen. De trainingen en duikexcursies worden gegeven aan de eigen leden van de club. De trainingen kunnen in het eigen zwembad worden gegeven, maar er zijn ook excursies naar externe locaties. De club gebruikt daarvoor in ieder geval de volgende gegevensbronnen:

- Ledentabel
- Trainingstabel
- Locatietabel
    - Contributietabel

De Duikwinkel

De winkel verkoopt duikartikelen aan personen. Dat kunnen leden van de duikclub zijn, die een standaard korting krijgen, of externen. Om de winkel te kunnen laten functioneren, zijn zeker de volgende tabellen nodig:

- Artikelentabel
- Facturentabel

- Leverancierstabel
  - Bestellingentabel

Het verhuurcentrum

De belangrijkste activiteit van een verhuurcentrum is het verhuren van duikspullen aan zowel leden, die korting krijgen, als externen. Daarbij worden duikspullen uit eigen voorraad verhuurd. Het verhuurcentrum gebruikt daarvoor tabellen als:

- Personentabel
- Materiaaltabel
  - Verhuurtabel

De tabellenstructuur is hiermee bij lange na niet compleet. Zo zal in ons systeem de duikwinkel facturen willen maken voor de goederen die worden verkocht. Deze facturen worden uiteraard opgeslagen.

Opzet van de database

De database wordt gebruikt door verschillende persoonsgroepen:

- Administratie

Deze groep houdt de ledenadministratie bij, en de duikknipkaarten/trainingen/excursies.

- Trainers

De trainers houden de trainingsresultaten bij, en verzorgen de certificatie van de leden.

- Verkoop en Verhuur

De verkoopafdeling houdt de verhuuradminstratie bij, en verkoopt duikartikelen. Tevens kopen zij materialen in voor de winkel en de verhuurafdeling.

- Serviceafdeling

De serviceafdeling onderhoudt de apparatuur. Dit geldt voor zowel de verhuurspullen, als de verkochte artikelen.

Elke groep moet toegang hebben tot verschillende onderdelen van de database. De serviceafdeling moet bijvoorbeeld gebruik maken van de magazijngegevens, en de artikelentabellen. Ook moeten zij producten kunnen koppelen aan der personen. Zij hoeven echter geen toegang te hebben tot de cursusadministratie. Het omgekeerde geldt voor de trainers. Zij hoeven niet bij de leverancierstabellen en voorraadgegevens, maar moeten wel weer kunnen muteren op de cursusgegevens.

Er moeten dus verschillende interfaces worden gebouwd voor de verschillende gebruikersgroepen. Dit houdt ook in, dat er een inlogsysteem moet komen waarmee een gebruiker binnen het systeem wordt geïdentificeerd, zodat de juiste interface wordt gestart.

Gescheiden Frontend en Backend

Om zoveel mogelijk personen tegelijk in de database te laten werken, is het wenselijk om de tabellen te scheiden van de gebruikersinterface. Dat doen we met een scheiding van de database in een *Frontend* database, en een *Backend* database. In de Backend database worden alle tabellen gemaakt, terwijl de frontend alle formulieren en noodzakelijke code bevat om deze optimaal te laten werken. Door middel van *Koppelingen* worden de Frontend en de Backend met elkaar verbonden, zodat de Frontend alle noodzakelijke tabellen kan gebruiken. Een frontend database voor de Trainersgroep kan dus andere tabellen bevatten dan de Frontend van de administratie.

Om de werking van de database verder te optimaliseren, wordt gekozen voor een onafhankelijke frontend, waarbij er alleen een koppeling naar de backend database wordt gemaakt als er gegevens moeten worden opgehaald of opgeslagen. Door deze minimale belasting op de backend wordt een optimale prestatieindex gehaald voor zowel de frontend als de backend.

De ledenadministratie

Het gegevensbestand van de duikclub bestaat uit een *ledenadministratie*, die is gebaseerd op een Lidmaatschap. Hierbij wordt onderscheid gemaakt tussen individuele abonnementen, en familie-abonnementen.

Een ledenadministratie kan zijn gebaseerd op personen, of op lidmaatschapsvormen. Met een persoonsgebonden systeem is het lastig om onderlinge relaties tussen de leden vast te leggen en te onderhouden: als twee leden met elkaar getrouwd zijn, moet dit gegeven bij beide personen worden ingevoerd. Dat leidt tot dubbele handelingen, en kan fouten tot gevolg hebben. Ook is het lastig om kinderen te koppelen aan de ouders, omdat niet bekend is of er één of twee ouders lid zijn van de club. Het kan ook lastig zijn om één exemplaar van de maandelijkse nieuwsbrief naar één adres van een gezin te sturen, waardoor er onnodig veel exemplaren worden gedrukt en verstuurd.

Omdat duiken een gezinsactiviteit is en een duikclub vaak meerdere leden uit één gezin bevat, kan de administratie ook worden opgezet vanuit één lidmaatschapskaart. Hierbij kan een groepslidmaatschap worden gekozen, waaraan meerdere personen kunnen worden toegevoegd. Voor de nieuwsbrief heeft dit als voordeel dat er maar één exemplaar kan worden verstuurd. Tevens kun je verschillende tarieven voor verschillende gezinssamenstellingen gebruiken. Hoe meer leden op de gezinskaart, hoe meer korting bijvoorbeeld. Vanwege deze voordelen voor de duikclub wordt gekozen voor een Groepen benadering.

De Duikwinkel

Zoals gezegd, worden in de winkel van het duikcentrum duikspullen verkocht. Voor deze artikelen moet uiteraard voorraad worden bijgehouden, en dat betekent een Voorraad- en besteladministratie. Bestellingen worden uiteraard bij leveranciers gedaan, die ook in de database moeten worden ingevoerd en bijgehouden. Per leverancier moeten we bijhouden welke artikelen ze verkopen, met welk artikelnummer dit moet worden besteld, en wat de prijzen zijn voor die artikelen. Ook moet worden bepaald of we de geschiedenis van de artikelprijzen willen bijhouden, of dat we de prijs bij elke prijswijziging doorvoeren in de tabel Artikelen. De eerste optie houdt de administratie inzichtelijker, en historisch correcter, maar vereist een heel andere aanpak dan de tweede die makkelijker is bij te houden.

De verkopen worden geregistreerd in een eigen tabel, waarbij we de koper een aankoopbon geven. Op deze bon staan de aangeschafte artikelen, met de op dat moment geldende verkoopprijs. Deze gegevens moeten ook worden opgeslagen in tabellen.

Het Verhuurcentrum

Voor het verhuren van duikspullen geldt, dat we per verhuurbaar artikel meerdere exemplaren in het magazijn zullen hebben. Bij het verhuren moeten we vastleggen wanneer een artikel aan een persoon is verhuurd, en wanneer het weer wordt ingeleverd. We moeten uiteraard voorkomen dat een artikel twee keer tegelijk kan worden uitgeleend, wat fysiek onmogelijk is en dus in de database moet worden afgedwongen. Ook voor de verhuur hebben we facturen nodig, en huurovereenkomsten.

Wat is een relationele database?

Een database beschouwen we als een relationele database, als we *verbanden* kunnen leggen tussen de onderlinge tabellen. Hierbij worden twee tabellen aan elkaar gekoppeld op basis van een *Kandidaatsleutel*, en een *Verwijzingssleutel*.

Begrippen binnen een database

Binnen een database worden andere termen gebruikt dan in bijvoorbeeld Excel, waarin je een tabel kunt maken die bestaat uit cellen, rijen en kolommen. Daarbij staat in één cel een bepaald gegeven, dat samen met andere gegevens cellen in dezelfde rij bij elkaar hoort, en op kolomniveau gegevens laat zien die allemaal van hetzelfde type zijn.

	A	B	C	D	E	F	G	H	I
1	LidMaatschapsID	Voornaam	Tussenvoegsel	Naam	Straatnaam	Nr	Toevoeging	Postcode	Stad
2	52	Raf		Ponjaert	Minstraat	32		3582 CD	Utrecht
3	71	Ed		Verstraete	W. Barendstraat	51		3572 PD	Utrecht
4	75	Ingrid		Willems	Gisbetlaan	32		3571 AK	Utrecht
5	117	Peter		Rosseneu	Oudwijkerveldstraat	2		3581 JL	Utrecht
6	176	Erwin	van	Waeleghem	Mercuriusbrug	13	D	3437 GX	Nieuwegein
7	177	Christel		Slegtinck	Grutto	14		3941 NM	Doorn
8	178	Kris		Maes	Biltstraat	152		3572 BN	Utrecht
9	179	Nancy		Monfrans	Schermerslaan	99		3705 GM	Zeist
10	180	Fernand		Vandendorpe	Oudwijk	21		3581 TG	Utrecht
11	181	Ronald		Verlinde	Jacob Catsstraat	134		2274 GZ	Voorburg
12	182	M-Claire		Tanghe	Hermannus Elconiusstr.	32		3553 VE	Utrecht
13	183	Patrick		Tytgat	Van Lieflandlaan	24		3571 AB	Utrecht
14	184	Paul	Vander	Cruysse	Burg. F. Andrealn	60	K	3582 KT	Utrecht
15	185	Carine		Decapmaker	Zaagmolenkade	39		3515 AD	Utrecht

In dit voorbeeld bevat de gele rij met rijnummer 9 de gegevens van één persoon, terwijl kolom D alleen achternamen bevat. Deze structuur wordt in een database ook gebruikt.

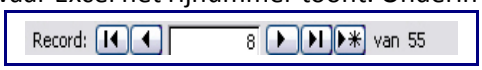
In een database wordt een rij gegevens die bij elkaar horen, zoals in het voorbeeld de gegevens in rij 9 het adres vormen van Nancy Monfrans, een *Record* genoemd. Een term die ook voor een record wordt gebruikt, is *Tupel*. Gegevens die in het voorbeeld in kolom C staan bevatten allemaal een Voornaam. Deze groep gegevens wordt in cel C1 benoemd; hier staat een kolomnaam, namelijk Voornaam. In een database noemen we kolommen *Veldnamen*. Merk op, dat de Rijen niet apart benoemd worden; in een database wordt het geheel benoemd, dus in het voorbeeld zou het blok A1:J12 een tabel kunnen zijn die tPersonen heet. In de tabel tPersonen vind je dus de adressen terug van de verschillende personen.

De Gegevensbladweergave

In een database ziet een gegevensblok, zoals bovenstaande Excel tabel er als volgt uit:

	LidMaatschaps	Voornaam	Tussenvoegsel	Naam	Straatnaam	Nr	Toevoeging	Postcode	Stad
+	52	Raf		Ponjaert	Minstraat	32		3582 CD	Utrecht
+	71	Ed		Verstraete	W. Barendstraat	51		3572 PD	Utrecht
+	75	Ingrid		Willems	Gisbetlaan	32		3571 AK	Utrecht
+	117	Peter		Rosseneu	Oudwijkerveldstraat	2		3581 JL	Utrecht
+	176	Erwin	van	Waeleghem	Mercuriusbrug	13	D	3437 GX	Nieuwegein
+	177	Christel		Slegtinck	Grutto	14		3941 NM	Doorn
+	178	Kris		Maes	Biltstraat	152		3572 BN	Utrecht
+	179	Nancy		Monfrans	Schermerslaan	99		3705 GM	Zeist
+	180	Fernand		Vandendorpe	Oudwijk	21		3581 TG	Utrecht
+	181	Ronald		Verlinde	Jacob Catsstraat	134		2274 GZ	Voorburg
+	182	M-Claire		Tanghe	Hermannus Elconiusstr.	32		3553 VE	Utrecht
+	183	Patrick		Tytgat	Van Lieflandlaan	24		3571 AB	Utrecht
+	184	Paul	Vander	Cruysse	Burg. F. Andrealn	60	K	3582 KT	Utrecht
+	185	Carine		Decapmaker	Zaagmolenkade	39		3515 AD	Utrecht

Je ziet, dat de indeling bijna identiek is aan die van het Excel blok. Alleen zijn de kolomletters en de rijnummers verdwenen. In Access wordt het geselecteerde record aangeduid door een driehoekje ()

waar Excel het rijnummer toont. Onderin het scherm vind je het *Navigatiepaneel*
(

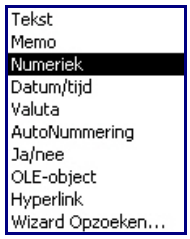
).

In het navigatiepaneel vind je knoppen waarmee je naar het begin van de tabel kunt bladeren, naar het vorige record, naar het volgende record, naar het laatste record en een Nieuw record. Als laatste zie je een aanduiding van het aantal records in de tabel.

De Ontwerpweergave

	Veldnaam	Gegevenstype
▶	LidNr	AutoNummering
	LidMaatschapsID	Numeriek
	Voornaam	Tekst
	Tussenvoegsel	Tekst
	Naam	Tekst
	Straatnaam	Tekst
	Nr	Tekst
	Toevoeging	Tekst
	Postcode	Tekst
	Stad	Tekst

Elke tabel begint in de Ontwerpweergave. Hierin bepaal je welke velden je gaat gebruiken, welke naam ze krijgen, en vooral: welk gegevenstype je er in gaat opslaan. Zo'n ontwerpscherm ziet er in het voorbeeld zo uit:



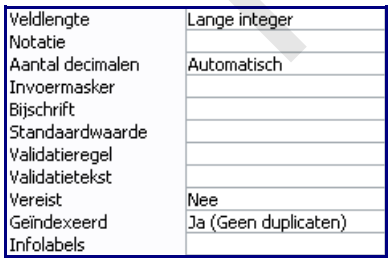
Je ziet dezelfde namen terugkomen die in de Gegevensweergave boven de gegevensrijen staan. Deze staan in de kolom *Veldnaam*. Je ziet de feitelijke waarden echter niet; daarvoor gebruiken we de Gegevensbladweergave. Wel zie je een extra kolom, met de naam *Gegevenstype*. Hiermee bepalen we welk soort gegeven we gaan opslaan in het veld. Er zijn verschillende gegevenstypen die je kunt toewijzen aan een veld. Deze vind je in bijgaande afbeelding.

Gegevenstype en Veldeigenschappen

Van elk gegevenstype zijn verschillende eigenschappen in te stellen. Soms is dat een eigenschap die voor alle gegevenstypen geldt, en soms is dat een eigenschap die specifiek is voor dat gegevenstype. Zo kun je voor de meeste velden een *Notatie* opgeven, maar alleen bij een tekstveld een *Lengte nul toestaan*. Ik laat dat zien aan de hand van twee velden.

Veldeigenschappen van een Numeriek veld

Om te beginnen het veld [LidMaatschapsNr]. Zoals je in bovenstaande afbeelding kunt zien, heeft dat veld het gegevenstype *Numeriek*. Dat houdt in dat we alleen getallen kunnen invullen in dit veld. Je kunt verschillende soorten getallen kiezen, die allemaal hun eigen karakteristieken hebben.
Access geeft onder het raster met de veldnamen een blok met verdere eigenschappen van het geselecteerde veld. Hierin kun je dus een Notatie instellen, het Aantal decimalen, een invoermasker etc.



Veldlengte	Lange integer
Notatie	
Aantal decimalen	Automatisch
Invoermasker	
Bijschrift	
Standaardwaarde	
Validatieregel	
Validatietekst	
Vereist	Nee
Geïndexeerd	Ja (Geen duplicaten)
Infolabels	

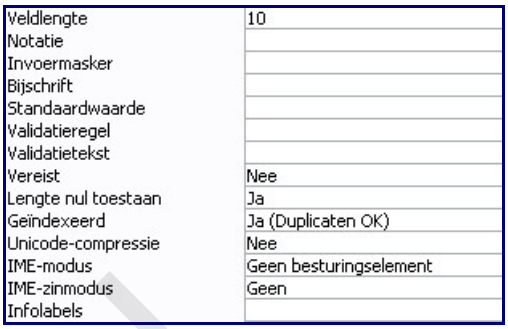
Een Notatie zou bijvoorbeeld kunnen zijn: **0.000**. Hiermee geef je aan dat elk getal wordt weergegeven met minstens 4 cijfers voor de getallen, met voorloopnullen als het getal uit minder dan 3 of minder cijfers bestaat. In dit voorbeeld ga ik er van uit dat het scheidingsteken voor duizendtallen een punt is. Gebruik je de punt als decimaalteken, dan wordt elk getal weergegeven zoals het is ingevoerd, met 3 decimalen.

Elk type veldlengte gebruikt een bepaalde hoeveelheid opslagruimte in de database. Die ruimte bepaalt hoe groot het getal kan zijn in de tabel. Een getal van het type *Byte* gebruikt de minste ruimte, maar je kunt er alleen getallen in opslaan tussen de 0 en 255. Het type *Integer* kan al getallen opslaan van –32.768 tot en met 32.767. En zo zijn er dus nog een aantal veldlengtes mogelijk. In de bijlage vind je een overzicht van alle veldlengtes. Als je in de regel *Veldlengte* klikt, en je drukt op <F1>, dan krijg je een overzicht van alle beschikbare veldlengtes.

opmerking Bij een veld met het gegevenstype *Autonummering* zie je dat de *Veldlengte* ook *Lange integer* is. Dit is (naast het minder gebruikt type *Replicatie-id*) de enige veldlengte die je kunt kiezen bij een Autonummering veld.

Veldeigenschappen van een tekstveld

Kijken we naar een veld als [Postcode] dan vind je bij de veldeigenschappen andere instellingen dan bij een Numeriek veld, omdat [Postcode] een tekstveld is.



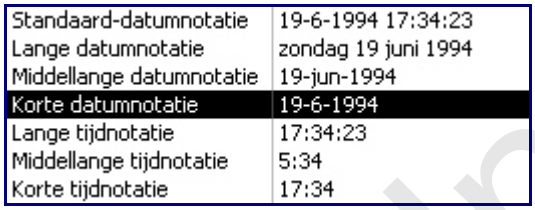
Veldlengte	10
Notatie	
Invoermasker	
Bijschrift	
Standaardwaarde	
Validatieregel	
Validatietekst	
Vereist	Nee
Lengte nul toestaan	Ja
Geïndexeerd	Ja (Duplicaten OK)
Unicode-compressie	Nee
IME-modus	Geen besturingselement
IME-zinmodus	Geen
Infolabels	

Hier wordt de veldlengte ingesteld op basis van het aantal *tekens* dat je op wilt slaan. In het voorbeeld staat de veldlengte ingesteld op 10 karakters.

Dat betekent, dat een tekst van 12 karakters zal worden afgebroken, omdat er maar plaats is voor 10 tekens. Een Tekstveld kan maximaal 255 tekens bevatten.

Eigenschap Veldnotatie

Bij een Tekstveld kun je een *Notatie* vastleggen; ongeveer op de manier waarop dat ook in Excel mogelijk is. Je kunt bijvoorbeeld aangeven dat een tekst altijd in hoofdletters moet worden getoond, of juist altijd in kleine letters. Een veldnotatie *heeft geen invloed* op de manier waarop de gegevens worden opgeslagen. Als de notatie van het veld staat ingesteld op Hoofdletters, dan worden de veldwaarden Utrecht, utrecht en UTRECHT allemaal weergegeven als UTRECHT.



Standaard-datumnotatie	19-6-1994 17:34:23
Lange datumnotatie	zondag 19 juni 1994
Middellange datumnotatie	19-jun-1994
Korte datumnotatie	19-6-1994
Lange tijdnotatie	17:34:23
Middellange tijdnotatie	5:34
Korte tijdnotatie	17:34

Een datumveld als [Geboortedatum] krijgt doorgaans het gegevenstype *Datum/Tijd*. Bij zo'n veld kun je verschillende soorten datumnotatie gebruiken. Naast de standaard notatie die je uit de keuzelijst kunt kiezen, kun je ook zelf een notatie instellen door de code voor die notatie in te typen.

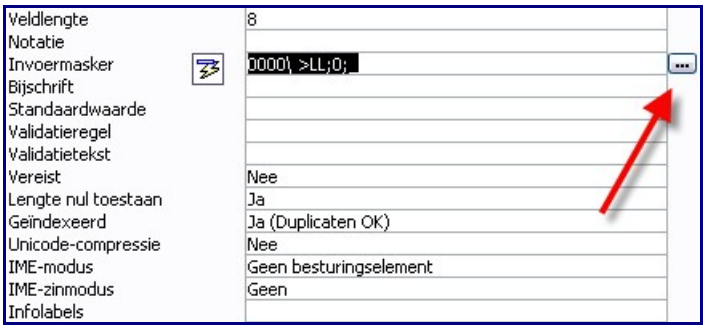
In het veld wordt een datum altijd opgeslagen inclusief een tijd; door de notatie te veranderen kun je aangeven of je de datum, de tijd of allebei wilt zien. Een veel gemaakte fout die mensen maken is, dat ze denken dat ze door de Notatie te veranderen ook de Veldeigenschappen veranderen. Door een veld de datumnotatie "yyyy" te geven zie je de datum als jaartal terug in de gegevensweergave. In de tabel wordt echter nog steeds de volledige datum (+tijd) opgeslagen.

belangrijk *Het is dus zaak om niet te vergeten dat een veldnotatie alleen het uiterlijk van het veld verandert, nooit de inhoud!*

Eigenschap Invoermasker

Soms kan het nuttig zijn om de gebruiker van de database te 'dwingen' om de gegevens volgens een van te voren vastgelegd patroon in te voeren. Denk bijvoorbeeld aan een veld voor postcodes, dat voor Nederlandse adressen altijd bestaat uit vier cijfers gevolgd door een spatie en twee hoofdletters. Voor dit soort vaste gegevens kun je een *invoermasker* maken. Met een invoermasker bepaal je welke tekens je op welke plek je moet invoeren.

Je kunt een invoermasker maken door op de knop met de drie puntjes te klikken, waarmee je een wizard start die je in een aantal stappen door het proces leidt. Voor het veld [Postcode] levert dit bijvoorbeeld het volgende invoermasker op: **0000\ >LL;0;_**



Als je de wizard hebt gestart, kies je een invoermasker, en loop je alle stappen door.

tip *Als je de cursor in een vakje achter één van de opties plaatst, en op <F1> drukt, krijg je specifieke hulp over de geselecteerde optie. Klik je bijvoorbeeld in de kolom Gegevenstype, dan krijg je een overzicht van de beschikbare gegevenstypes en hoe je ze kunt gebruiken. Klik je in het vakje Invoermasker, dan zie je welke opties je kunt gebruiken, en wat de verschillende tekens betekenen. Met deze kennis kun je uiteraard ook zonder de wizard zelf een invoermasker samenstellen.*

Een Notatie-opmaak is overigens niet hetzelfde als een Invoermasker. Met Notatie geef je alleen op hoe een veld moet worden weergegeven; met een invoermasker bepaal je heel specifiek wat een gebruiker mag intypen. Deze twee instellingen moet je dus niet met elkaar verwarren!

Eigenschap Standaarwaarde

Naast een afwijkende notatie en/of een invoermasker, kun je een veld een vooringestelde waarde meegeven. In de tabel [Verhuur] kun je bijvoorbeeld een standaard Verhuurdatum instellen. De gebruiker hoeft dan bij het uitlenen geen datum in te typen; deze staat dan al in het datumveld. Een standaardwaarde kan een vaste waarde zijn, of een formule. Voorbeelden van een standaardwaarde zijn:

Veld	**Veldtype**	**Standaardwaarde**
Plaats	Tekst	“Rotterdam”
Verhuurdatum	Datum/Tijd	=Date()
Contributie	Valuta	25

Eigenschap Validatieregel en Validatietekst

Access heeft de mogelijkheid om gegevens te controleren als een gebruiker een nieuw record toevoegt. Je kunt bijvoorbeeld instellen dat in het veld [Contributie] minimaal € 20,- moet worden ingevuld. Je vult dan in de regel *Validatieregel* de formule: **>20** in. Bij *Validatietekst* typ je dan de boodschap die verschijnt als de gebruiker een te lage waarde intypt. Bijvoorbeeld: “U moet minimaal € 20,- invullen.”

Eigenschap Vereist

Met de Eigenschap *Vereist* geeft je aan of een veld verplicht moet worden ingevuld of niet. Als deze eigenschap op **Ja** staat, kun je het record niet opslaan als het veld niet is ingevuld. Het lijkt verleidelijk om deze eigenschap bij alle velden op Ja te zetten, zodat je zoveel mogelijk gegevens opslaat. Bij veel velden kun je daarmee echter in de problemen komen. Een veld als [Tussenvoegsel] bijvoorbeeld, is lang niet op iedereen van toepassing. Als dit veld verplicht wordt gesteld, kun je geen record opslaan als er niets is ingevuld, en dat is uiteraard niet de bedoeling. Kies de velden die je verplicht maakt dus zorgvuldig uit!

Eigenschap Geïndexeerd

Een eigenschap die de werking van de database aanzienlijk kan verbeteren is de eigenschap *Geïndexeerd*. Als er een index op een veld is ingesteld, gaat het zoeken op dat specifieke veld aanzienlijk sneller dan als er geen index wordt gebruikt. Ook hiervoor geldt dat het verleidelijk is om zoveel mogelijk velden te indexeren. Maar net als bij de vorige eigenschap geldt: doe dit met mate; met teveel indexen in een tabel wordt de zoekfunctie juist trager i.p.v. sneller. Access moet dan op de achtergrond teveel handelingen uitvoeren om nog snel te kunnen werken. Een index kan op twee manieren worden ingesteld:

- Ja (Duplicaten OK)
  - Ja (Geen duplicaten)

Een duplicaat is in dit geval een identieke waarde in hetzelfde veld. Als de optie *Ja (Geen duplicaten)* wordt gebruikt voor het veld [Plaatsnaam], kun je een plaatsnaam maar één keer gebruiken. De volgende keer dat je de plaatsnaam invult, komt er een foutmelding van Access omdat de naam al eerder is gebruikt. Dit is meestal niet de bedoeling. Een normale indexering wordt dus gemaakt met de optie *Ja (Duplicaten OK)*.

Gebruik Indexering voor velden die veel repeterende waarden bevatten, zoals Plaatsnamen, Functies, Geslacht etc. Bij dit soort velden merk je, zeker bij grotere tabellen, een aanzienlijke snelheidswinst.

De wizard Opzoeken

Een vreemde eend in de bijt is de wizard *Opzoeken* die je kunt kiezen bij Gegevenstype. (Zie ook het plaatje op pagina 7). Hiermee kun je voor een veld een Keuzelijst maken, zodat een gebruiker een tekst niet hoeft in te typen, maar deze kan selecteren uit een keuzelijst met invoervak.

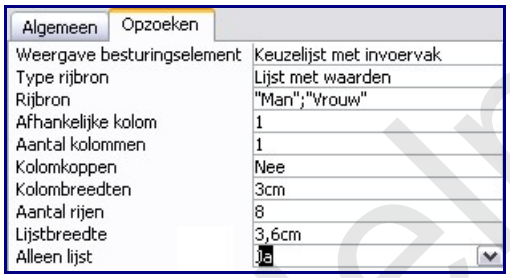
opmerking Dezelfde techniek komen we later ook tegen op formulieren.

Bij het maken van een keuzelijst kun je kiezen tussen een keuzelijst die is gebaseerd op een Tabel, of een keuzelijst die is gebaseerd op waarden die worden ingetypt. Aangezien ik van mening ben dat in een tabel *altijd* de werkelijke waarden te zien moeten zijn, gebruik ik de eerste optie nooit. Een lijst maken met vaste teksten vind ik daarentegen wel een zinvolle optie.
In dit voorbeeld maken we een keuzelijst voor het veld [Geslacht], waar, zoals bekend, maar een aantal opties te verzinnen zijn.

Als je de wizard opstart, kies je dus voor de optie *De waarden zullen worden getypt.*
Daarna klik je op **Volgende**

In de kolom typ je de opties waaruit je wilt kunnen laten kiezen, en daarna kun je de wizard afmaken door op de knop *Volgende* te klikken.

Door een keuzelijst te maken en in te stellen dat alleen waarden uit de keuzelijst mogen worden gebruikt, dwing je de gebruiker om een waarde te kiezen, en kunnen geen andere waarden worden getypt.



Als je het principe van een keuzelijst eenmaal kent, kun je deze ook zelf maken, door bij de Eigenschappen het tabblad *Opzoeken* te openen, en de optie *Weergave besturingselement* in te stellen op **Keuzelijst met invoervak**. Zoals je kunt zien, is de optie *Type rijbron* ingesteld op **Lijst met waarden**, en zijn de verschillende opties getypt bij de optie *Rijbron*.

Sleutels

Om in een database tabellen aan elkaar te kunnen relateren (we hebben het tenslotte over een relationele database) hebben we een manier nodig om één record uit een tabel te kunnen identificeren, en dat record terug te kunnen vinden in de gerelateerde tabel. Daarvoor gebruiken we het begrip *Sleutel*. Er zijn verschillende vormen van sleutels:

- Kandidaatsleutel
- Primaire sleutel
    - Verwijssleutel

In de volgende paragrafen worden deze sleutelmodellen behandeld. Het is de bedoeling om met behulp van de Primaire sleutels en Verwijssleutels tabellen aan elkaar te koppelen. We noemen dit de *Relatie* tussen de tabellen. Relaties worden in de volgende cursus behandeld.

Kandidaatsleutel

Binnen een tabel kan één record op een aantal manieren worden geïdentificeerd. Een eigenschap van een sleutel is dat je met een sleutel een record kunt opzoeken waarvan geen duplicaten in de tabel zijn te vinden. Dat kan soms met één veld, soms met een combinatie van velden. Binnen een tabel kun je vaak meerdere sleutels aanwijzen. Dit zijn dan allemaal Kandidaatsleutels. In een adressenbestand kun je bijvoorbeeld kandidaatsleutels maken van de velden [Straatnaam]+[Huisnummer}+[Plaats], en van [Postcode]+[Huisnummer].

Definitie *Een kandidaatsleutel is een minimale verzameling van velden die de verzameling records uniek identificeert.*

Primaire sleutel

Meestal hebben tabellen één kandidaatsleutel, maar soms zijn er meer kandidaatsleutels mogelijk. Per tabel kiezen we één sleutel als *primaire sleutel*. Voor de primaire sleutel kiezen we dan de kandidaatsleutel met het minste aantal velden. In het voorbeeld van de adressentabel maken we van [Postcode]+[Huisnummer] de primaire sleutel. De combinatie [Straatnaam]+[Huisnummer}+[Plaats] noemen we dan een *alternatieve sleutel*. In een tabel met persoonsgegevens zou het SOFI-nummer, of het BSN-nummer als primaire sleutel kunnen dienen.

opmerking Vaak wordt binnen een tabel een ID-veld gemaakt op basis van het gegevenstype *Autonummering*. Omdat deze nummers altijd uniek zijn, kan zo'n veld dus probleemloos als primaire sleutel worden gebruikt. Sleutels die op deze manier worden gemaakt zeggen echter weinig over het record dat ze identificeren. Het verdient dus aanbeveling om primaire sleutels te baseren op velden die een betekenis hebben voor de gebruikers.

Verwijssleutel

In de tabellen waarin we gegevens opslaan die te maken hebben met de kerntaken of activiteiten van de organisatie, zoals verkoopactiviteiten of trainingsschema's, moeten we de gegevens opslaan van de personen of bedrijven waarop die transacties betrekking hebben. Bij een trainingsactiviteit willen we bijvoorbeeld weten wie daar aan meegedaan hebben. In plaats van het invullen van de namen, adressen, telefoonnummers etc. van alle deelnemers, vullen we in die tabel liever het Ledennummer in. Dit ledennummer hebben we namelijk in de tabel Duikleden gedefinieerd als Primaire sleutel. Als we dus het ledennummer weten, dan kunnen we in de tabel Duikleden de overige gegevens van het lid opzoeken.

In de tabel Trainingen slaan we dus het veld [LidmaatschapsID] op bij de trainingsactiviteit, en niet de eerder genoemde losse gegevens. Dit voorkomt niet alleen dat we heel veel gegevens dubbel invoeren in de tabellen, maar ook dat er typfouten worden gemaakt bij het invoeren. Op basis van het ingevoerde Lidmaatschapsnummer kunnen we later m.b.v. queries wel weer de oorspronkelijke naam- en adresgegevens aan de training koppelen voor bijvoorbeeld de facturen of overzichten.

Samenvatting

In deze cursus gaan we een database maken voor een duikclub. Deze database wordt gesplitst in een Front-end en een backend database, waarbij in de Backend de gegevens worden opgeslagen, en in verschillende Frontend databases verschillende interfaces worden gemaakt voor de verschillende doelgroepen die met de database gaan werken.

In een Access database worden verschillende *objecten* gebruikt, zoals Tabellen, Formulieren en Rapporten. De database is gebaseerd op de tabellen; hierin worden de verschillende gegevens opgeslagen. Binnen de

verschillende tabellen proberen we de gegevens zodanig op te slaan, dat we deze zo min mogelijk hoeven in te voeren: we noemen dit *Gegevensredundantie*.

Bij het ontwerpen van een tabel kunnen we verschillende aspecten van de velden instellen, zoals het *Gegevenstype*, de *Notatiewijze* en een *Invoermasker.* Met validatieregels kunnen we afdwingen of een ingevoerde waarde voldoet aan een ingesteld criterium, en met een *Standaardwaarde* kunnen we een vast gegeven, of een functie instellen waarmee een veld een waarde krijgt, zodat de gebruiker dat niet hoeft in te vullen.

Een relationele database is een database waarin de tabellen via primaire sleutels en verwijssleutels aan elkaar worden gekoppeld. Binnen een tabel proberen we primaire sleutels te definiëren die een betekenis hebben voor de gebruiker, en die uit zo weinig mogelijk velden bestaan.

OPDRACHT

Een cursus zonder opdrachten is uiteraard geen cursus, maar een handleiding… Omdat het mijn opvatting is dat je in ieder geval moet begrijpen wat een database is, en uit welke elementen een goede database is opgebouwd, bevat deze eerste les relatief veel theorie. De eerste opdracht is in overeenstemming met de behandelde stof.

- Maak een model voor de database van het Duikcentrum. Maak in ieder geval díe tabellen waarvan jij vindt dat die nodig zijn in de database.

- Stel de veldeigenschappen zodanig in dat deze overeenkomen met de gegevens die je in die velden wilt opslaan. Dat houdt bijvoorbeeld in dat je een Postcodeveld geen lengte geeft van 50 tekens, omdat een postcode niet meer dan 7 karakters zal beslaan.

- Maak, daar waar mogelijk, invoermaskers om de in te voeren waarden te controleren.
Gebruik daar waar mogelijk ook standaardwaarden voor velden.

- Vul de tabellen met een aantal records, om te controleren of de gegevens op de juiste manier zijn in te vullen. Verbeter de tabellen als dat nodig is.

Volgende Aflevering

In de volgende lessen zal deze theorie veel minder aan bod komen, maar zal er wel regelmatig worden verwezen naar de stof die in dit hoofdstuk is behandeld.

In de volgende aflevering behandelen we eerst het principe van het Normaliseren van de database, waarbij de database naar de 3^{e} normaalvorm wordt gebracht.

Daarna worden de Relaties tussen de verschillende tabellen gelegd. Hierbij wordt de nadruk gelegd op het afdwingen van *Referentiële Integriteit*.

Index

[i] Redundantie betekent ‘Overtolligheid’ Het houdt in, dat in een tabel een bepaald gegeven meerdere malen moet worden opgeslagen. Door redundantie te vermijden wordt de database betrouwbaarder, en makkelijker te onderhouden.