Access voor Beginners - Hoofdstuk 7

Handleiding van Helpmij.nl

Auteur: Octafish

Augustus 2011

Een thema waar ik (redelijk bewust) omheen ben gelopen tot nu, is het onderwerp *Queries*. Niet omdat ik queries niet belangrijk vind, of overdreven moeilijk, maar simpel omdat ze tot nu nog niet noodzakelijk waren voor de cursus. Maar dat gaat dus veranderen!

Wat zijn queries?

Een op het oog simpele vraag, maar zoals wel vaker met simpele dingen: er zit een diepgang in die je op het eerste gezicht niet zomaar ziet. Queries worden namelijk gebruikt om gegevens op te vragen, gegevens te manipuleren, gegevens aan te maken, gegevens te berekenen of zelfs gegevens te verwijderen. En je komt ze op allerlei verschillende plekken tegen in de database: als zelfstandig object, als bron voor een formulier of rapport, en als bron voor keuzelijsten. Kortom: queries kom je overal tegen!

Om te beginnen maar eens een (arbitraire) definitie van een query: Een query is een instructie om bewerkingen op een database uit te voeren. En die query gebruikt daarvoor een eigen taal: SQL.

Om een lang verhaal kort te maken: als je het neusje van de zalm van Databases wilt weten, dan zul je je moeten verdiepen in queries, en daarom zul je dus ook de bijbehorende taal moeten leren: SQL. En omdat SQL door Amerikanen is vastgelegd, zal het je niet verbazen dat de voertaal voor queries het Engels is. Het woord ‘query’ zelf betekent overigens: vraag.

Wat is SQL?

SQL staat voor: **S**tructured **Q**uery **L**anguage. En dat is een taal die al sinds 1986 als standaard geldt voor Relationele Databases. De taal is relatief simpel opgebouwd: we hebben een aantal standaardbegrippen, waarmee we de query opbouwen. Afhankelijk van de functie van de query, voegen we daar nog wat extra commando’s aan toe. We zullen de belangrijkste begrippen in dit hoofdstuk uitgebreid aan bod laten komen.

Welke soorten queries zijn er?

Zoals ik hierboven al aangaf, kun je met queries verschillende handelingen uitvoeren, zoals gegevens bijwerken, records verwijderen en gegevens toevoegen. Daarmee hebben we al gelijk een groot onderscheid te pakken, puur op gebruiksdoel.

1. Selectie query
2. Bijwerk query
3. Toevoeg query
4. Verwijder query
5. Kruistabel query
6. Tabelmaak query

Daarnaast zijn er nog een paar buitenbeentjes:

1. Definitie query
2. Samenvoeg query

Maar laten we eens beginnen met het meest voorkomende type: de selectie query. Daar is op zich al dermate veel over te vertellen, dat we de andere types graag in een ander hoofdstuk onder de loep nemen!

De Selectie query

Zoals we inmiddels weten, slaan we onze gegevens op in verschillende tabellen. Als je een bepaald gegeven zoekt, kun je uiteraard de tabel openen, en door de records bladeren tot je het gewenste record hebt gevonden. Maar we kunnen ook een query maken die het gegeven voor ons opzoekt, en op het scherm toont.

Een selectie query gebruiken we om *gegevens uit een tabel te selecteren*. Het resultaat van een selectiequery is altijd een tabel, en wel een *dynamische tabel*. Dynamisch, omdat het resultaat van de query afhankelijk is van de gegevens die in de tabel staan. Dat klinkt vrij logisch als je dat zo leest; toch is alleen al het type Selectiequery een eigen hoofdstuk waard. En dat komt, omdat je heel veel verschillende dingen kunt doen met een selectiequery. De essentie van een selectiequery bestaat maar uit 4 woorden: SELECT, FROM, WHERE en ORDER BY. Hiermee wordt de hele query opgebouwd.

Zoals gezegd: query betekent: 'vragen'. Een query stelt dus een 'vraag' aan de database. En die database geeft daarop antwoord middels een tabel. Bij het ontwikkelen van een query kun je dan ook het beste beginnen met voor jezelf de vraag die je wilt beantwoorden in je eigen taal te stellen. Goede kans dat je syntax (de regels waaraan de query moet voldoen) dan al bijna gevonden hebt: alleen nog vertalen naar het Engels…

Bijvoorbeeld de vraag: welke leden van de duikclub zijn deze maand jarig? De gegevens van de leden vinden we in de tabel [st_Leden], en uit die tabel hebben we dan de velden [Naam] en [Geboortedatum] nodig.

De algemene syntax van een selectiequery is:

SELECT … FROM … WHERE … ORDER BY …

Oftewel: selecteer [typ vervolgens je veldnamen] uit [de tabel] waar [hier komt de beperking op de gegevens] [en sorteer ze vervolgens op een veld]

Merk op dat het niet noodzakelijk is om een query te sorteren; vaak zul je dat wel doen om de gegevens overzichtelijker te kunnen presenteren.

Om de syntax los te laten op de vraag naar de jarigen van deze maand:

SELECT [Naam], [Geboortedatum] FROM [St_leden] WHERE Month([Geboortedatum]) = Month(Date()) ORDER BY [Geboortedatum]

Als we de query opdelen in de hoofdelementen, dan ziet dat er zo uit:

SELECT [Naam], [Geboortedatum]

We willen twee velden zien: Naam en geboortedatum. Je kunt alle velden uit een tabel opvragen in een query, zolang ze maar gescheiden zijn door een komma. Met het commando SELECT * vragen we alle velden uit de tabel op; dat scheelt uiteraard een hoop typewerk!

FROM [St_leden]

Met FROM geven we aan uit welke tabel we de velden opvragen. In dit voorbeeld is dat één tabel; we zullen later zien dat je ook meer tabellen kunt gebruiken in een query, wat queries extreem veelzijdig maakt.

WHERE Month([Geboortedatum]) = Month(Date())

Dit deel is het lastigst; hier zit namelijk een extra functie bij, die ik er stiekem heb bijgesleept. Ik wilde in de oorspronkelijke vraagstelling de jarigen weten van deze maand. Normaal gesproken zeg je dan: iedereen die jarig is in januari, maart, augustus etc. Alleen: in de tabel heb ik dat gegeven niet; ik heb alleen een geboortedatum. Uiteraard weet je in welke maand iemand is geboren als je de geboortedatum weet. En Access heeft speciale Datum/Tijd functies, die je kunt gebruiken om datums te manipuleren. Eén van die functies is de functie MONTH, die de maand berekent uit een datum. Een vergelijkbare functie is YEAR, die (het zal je niet verbazen) het jaartal uit een datum haalt. En om dit type functie compleet te maken: met DAY lees je de dag uit van het datumveld.

De WHERE vergelijking kijkt dus naar de maandwaarde van de geboortedatum, en vergelijkt die met de maandwaarde van een andere datumfunctie die ik heb gebruikt: de functie DATE(). En die berekent (ook weer totaal niet verrassend) de huidige datum. De formule zegt dus letterlijk: WAAR

de MAAND van de geboortedatum gelijk is aan de MAAND van VANDAAG.

Het laatste deel van de query (ORDER BY) wordt gebruikt om het resultaat te sorteren op het veld Geboortedatum. Zoals je waarschijnlijk wel weet, zijn er twee vormen van sorteren: oplopend en aflopend. Dat geldt uiteraard ook voor de opdracht Order By. De standaard van dit commando is Oplopend sorteren. Wil je dat, dan hoef je verder niks aan te geven. Wil je echter aflopend sorteren, dan voeg je achter het sorteerveld het woord DESC toe. In het voorbeeld wordt het dan: ORDER BY [Geboortedatum] DESC. Wil je specifiek aangeven dat je oplopend wilt sorteren, dan zet je ASC achter het sorteerveld.

Net als bij het kiezen van velden, kun je ook voor de sortering meerdere velden aangeven. Hierbij wordt het eerste veld als eerste gesorteerd, gevolgd door het tweede, en zo verder. Je kunt ook oplopend en aflopend door elkaar gebruiken bij de verschillende sorteervelden.

Een query maken in Access

Nu we weten hoe een query is opgebouwd, en we van de eerste schrik zijn bekomen, wordt het tijd om zelf de handen uit de mouw te steken. Gelukkig heeft Access een grafisch scherm waarin we queries kunnen maken. Als je daar geen zin in hebt, zul je zelfs zelden of nooit iets hoeven te typen in je query: je kunt de meeste tabellen/velden/functies uit lijsten halen!

Omdat deze cursus bedoeld is om inzicht te geven in de onderliggende theorieën, heb ik er voor gekozen om eerst uit te leggen hoe eenvoudig een query eigenlijk is te maken, door simpel een aantal commando's achter elkaar te zetten. Maar nu dus het grafische geweld!

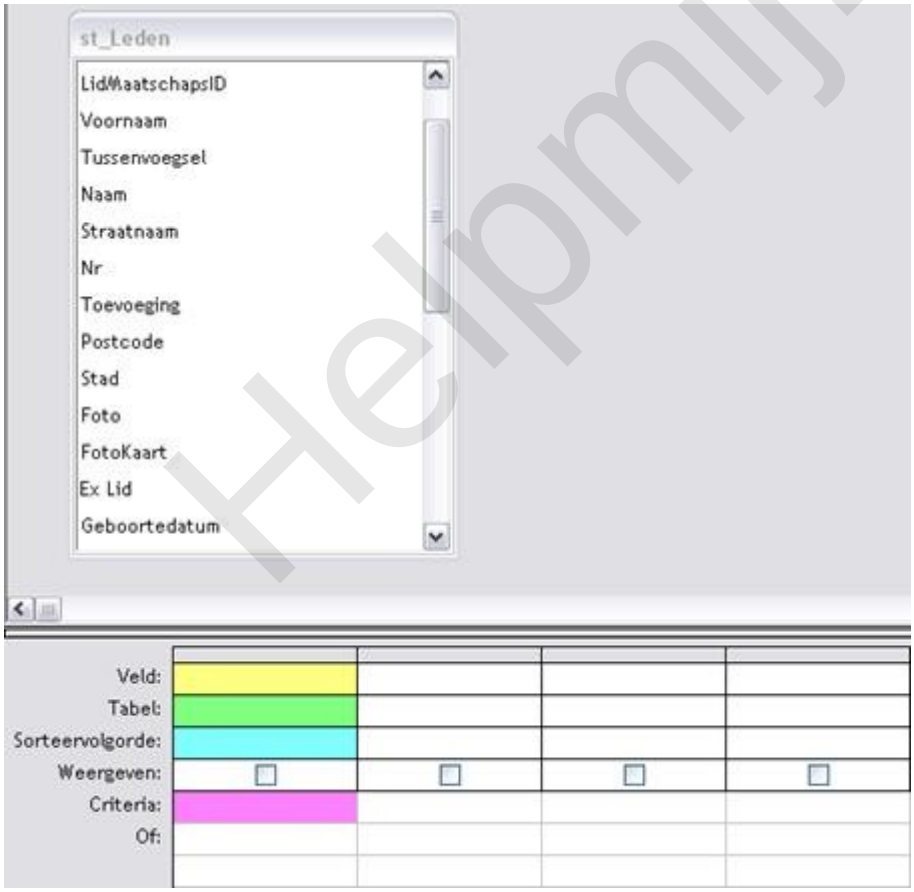
- In Access 2003 maak je een query door eerst op de groep <Query's> te klikken, en daarna op de knop <Nieuw>
- In Access 2007/2010 klik je op de groep <Maken>, en daarna op de knop <Query-ontwerp>

Je krijgt een venster te zien, waarin je kunt aangeven wat voor type query je wilt maken.

- Kies uit de lijst: **Ontwerpweergave** en klik op **OK**

- Selecteer de tabel die je gaat gebruiken, klik op **Toevoegen** en daarna op **Sluiten**

Je komt nu in het Query-raster; hier zie je de geselecteerde tabel staan, en daaronder een aantal lege rijen.



Het scherm hierboven is genomen uit Access 2003; in Access 2007/2010 heeft Microsoft bedacht dat het een stuk overzichtelijker is als alle lijnen lichtgrijs zijn… Laat ik over die beslissing maar geen mening geven, maar het laten bij de opmerking dat ik na een paar maanden met Access 2007/2010 gewerkt te hebben voor mijn queries nog steeds teruggrijp naar Access 2003, als ik de kans krijg J. De leesbaarheid van het raster is er in mijn ogen in ieder geval niet op vooruit gegaan…

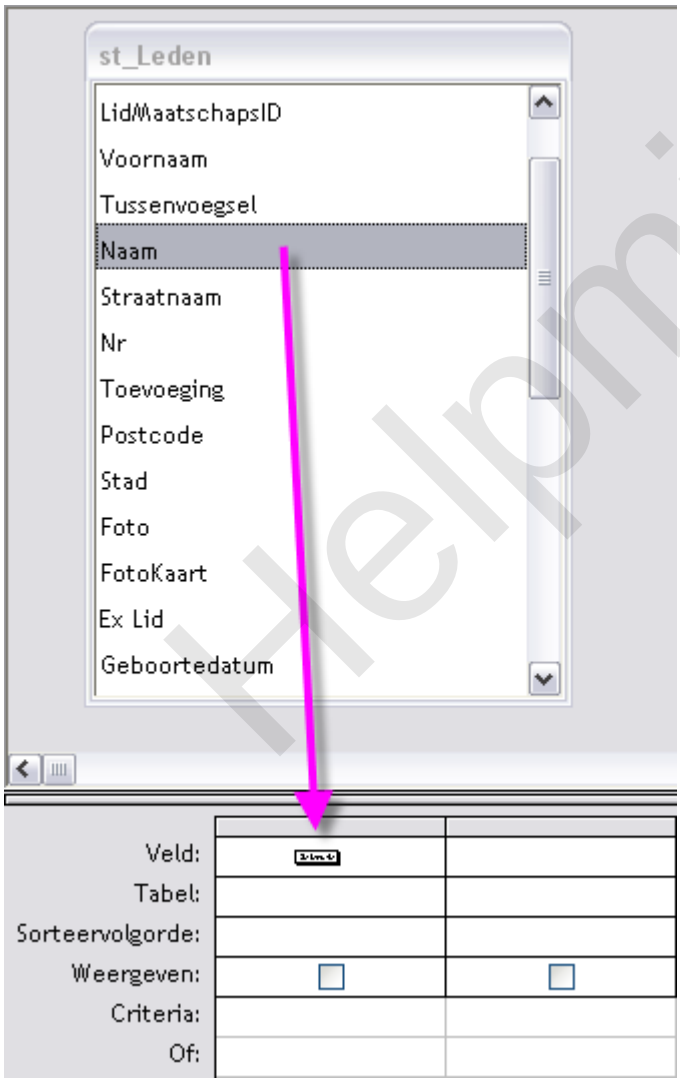
Overigens zijn de kleurtjes bedoeld om aan te geven wat je ziet op het scherm, en is dat (helaas)

geen instelling van Access.

De gekleurde vakjes geven aan wat je nodig hebt in deze query. Er is in een query minstens één veld nodig (gele veld), en dat veld komt (logisch…) uit een tabel (groene veld). Het blauwe veld bevat opties voor de sortering (oplopend, aflopend of geen), en de regel *Weergeven* bevat een selectievakje. Je kunt namelijk voor elk veld aangeven of je het wilt laten zien of niet. Het nut daarvan wordt later in de cursus wel duidelijk, mocht je daar nu over twijfelen.
Het paarse vak tenslotte is één van de rijen die je kunt gebruiken om het filter te maken.

1. Stap 1 is het selecteren van de gewenste velden. Daarvoor kun je verschillende technieken gebruiken:
    1. Je kunt een veld in de tabel selecteren, en daar op dubbelklikken
    2. Je kunt een of meer velden naar het onderste deel van het scherm slepen
    3. Je kunt in de rij *Veld* in een vakje klikken, en met de keuzelijst een veldnaam selecteren

Die laatste techniek zie je in bijgaand plaatje.



1. Dubbelklik op het veld **Geboortedatum** om dat ook toe te voegen aan het raster.

Nu er twee velden in de query zitten, kunnen we al kijken wat het resultaat is als we de query uitvoeren. Daarvoor hebben we een knop *Uitvoeren*. () Als je hier op klikt, zie je het resultaat van de selectie.

1. Klik op de knop **Uitvoeren** om het resultaat te bekijken.

	Naam	Geboortedatum
▶	Dhoore	1-11-1969
	Torre	1-5-1978
	Kokken	1-1-1981
	Velle	16-4-1992
	Schiltz	10-5-1972
	Boussy	4-6-1980
	Coppieters	25-1-1991
	Dandois	10-9-1993
	Decadt	4-11-1982
	Defever	19-9-1989
	Defever	3-8-1957
	Wybouw	10-5-1958
	Deneire	17-6-1984
	Damme	31-7-1984
	Dombret	1-4-1977
	Slimbrouck	16-11-1980
	Dombret	27-7-1947
	Dombret	5-10-1978
	Ameys	30-5-1995
*		

Je ziet de namen van alle leden, met hun geboortedatum.

1. Klik op de knop *Beeld* en kies **Ontwerpweergave** om weer terug te keren naar het ontwerpscherm.

We gaan nu het filter maken (je weet nog wel: het WHERE gedeelte)

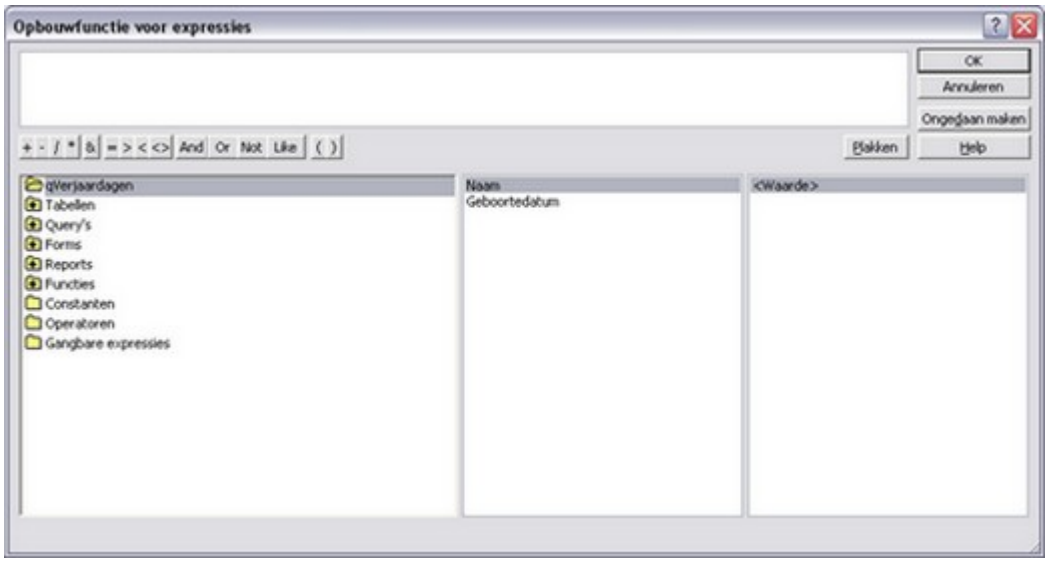
We doen dat met de *Opbouwfunctie voor Expressies*; hiermee kun je grafisch allerlei formules maken, waarbij Acces je helpt met de juiste syntax van die formules.

Tip: Het is verstandig om de query alvast op te slaan, zodat je de velden in de query kunt gebruiken.

1. Zet de muis in het eerste veld achter *Criteria* (het paarse veld uit het eerdere plaatje)

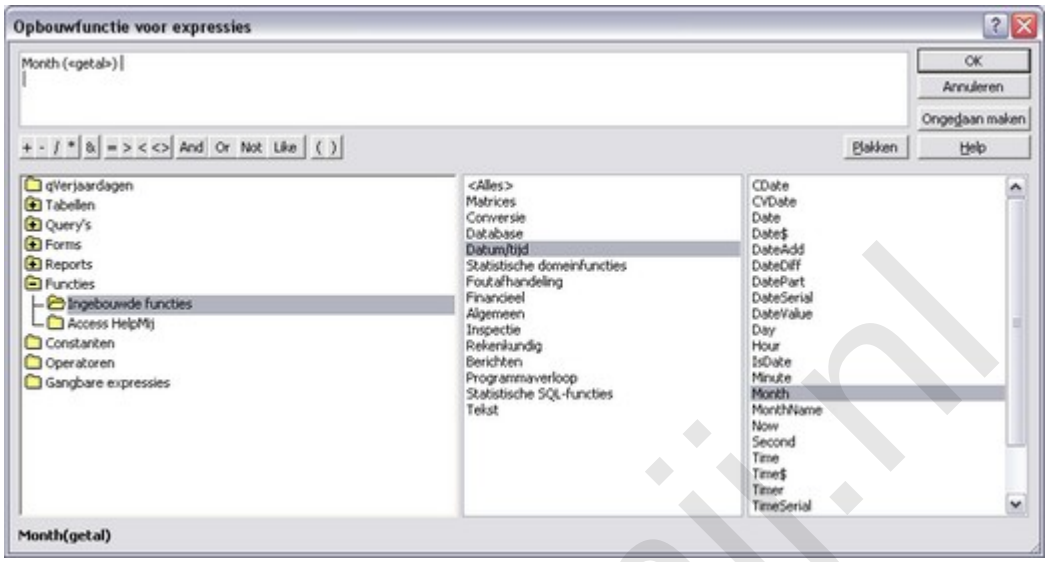
We gaan de vergelijking maken met de *Opbouwfunctie voor Expressies*. Daarvoor kun je met de rechtermuisknop in het Criteria veld klikken, of op de knop *Opbouwen.*

1. Klik op de knop **Opbouwen** ()
   Het venster *Opbouwfunctie voor Expressies* verschijnt.



Je kunt hier een expressie maken met de ingebouwde functies van Access. Deze functies zijn ondergebracht in de categorie Functies in het linker scherm.

1. Dubbelklik op het gele mapje of het woord *Functies* en klik op **Ingebouwde functies**
   De groep *Functies* wordt geopend in het middelste venster, in het rechter venster zie je de functies staan die bij de geselecteerde groep horen.
2. Dubbelklik op de functie **Month** uit de groep *Datum/Tijd*



De functie *Month* staat nu in het bovenste deel van het venster. Tussen de haakjes zie je de aanduiding «getal». Hiermee geeft Access aan dat de functie een waarde nodig heeft. In dit geval gaan we de maand berekenen van de geboortedatum.

1. Klik nu met de muis in het woord «getal» in het bovenste venster, en vervolgens op de naam van de query in het linker venster. Dubbelklik nu op de veldnaam **[Geboortedatum]**
   In het bovenste venster staat nu: *Month([Geboortedatum])*.

10. Typ of klik nu op het =-teken, selecteer uit de Datum/Tijd groep wederom de functie Month, selecteer wederom het woord «Getal», en dubbelklik nu in de groep met Datum/Tijd functies de functie **Date**

Als het allemaal goed is gegaan, staat er nu deze formule:

Month ([Geboortedatum]) = Month (Date ())

11. Klik op de knop **OK** om het venster af te sluiten, en de functie in het criterium veld te zetten.

12. Voer de query opnieuw uit.

Het resultaat zal nu afhangen van het aantal mensen dat in de geselecteerde maand jarig is. Omdat je elke maand een ander resultaat zult krijgen als je de query uitvoert, heet het resultaat een *Dynamische tabel*.

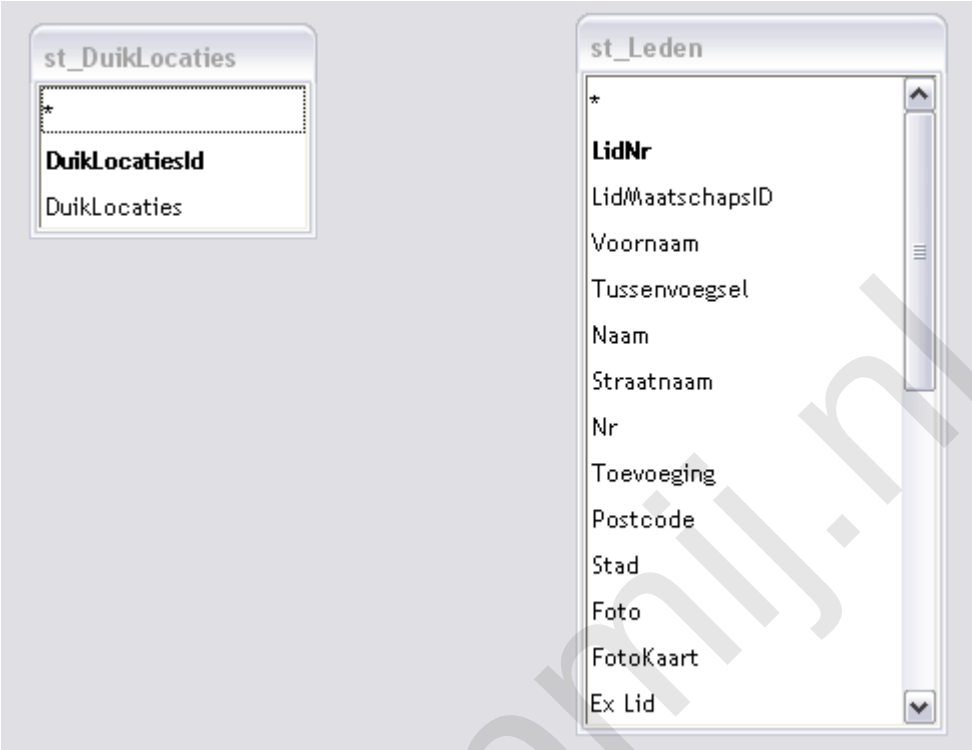
Opmerking: In de functie *Opbouwen* komt het vaak voor dat Access steekwoorden in de formules plaatst. Een veel voorkomend steekwoord is bijvoorbeeld «Expr», dat in de praktijk vaak niet nodig blijkt te zijn in de op te bouwen formule. Als je deze termen laat staan, zul je zien dat de functie niet werkt.

Queries op basis van meerdere tabellen

Je kunt queries maken op basis van meer dan één tabel. Daarmee kun je op bijna onbeperkte manier lijsten genereren van gegevens. Zo kun je bijvoorbeeld, als je een mailing wilt maken voor een excursie naar een bepaalde duiklocatie, een overzicht maken van alle leden die nog niet op een bepaalde plaats hebben gedoken. Deze leden krijgen dat bij het toekennen van (het beperkte aantal)

plaatsen voor die excursie voorrang.

In deze query nemen we alle velden op die in de uitnodiging moeten komen te staan. Dat zijn dus in ieder geval de Ledengegevens, het NAW blok uit de tabel. Maar uiteraard ook alle gegevens over de locatie. Deze gegevens staan in verschillende tabellen: de klantgegevens staan in de tabel [tblLeden], de locatiegegevens in de tabel [Duiklocaties].



Als we deze twee tabellen toevoegen aan het Query-raster, en we selecteren alle velden die we willen zien, en voeren vervolgens de query uit, dan zien we iets vreemds: i.p.v. een lijst van alle leden die al op de locatie hebben gedoken, zien we een lijst met veel meer records dan er leden of locaties zijn.

LidNr	Voornaam	Tussenvoegsel	Naam	Straatnaam	Nr	Postcode	Stad	DuikLocaties	DuikLocatiesId
266	Tine		Dhoore	Briljantlaan	142	3523 CJ	Utrecht	Oosterschelde	1
266	Tine		Dhoore	Briljantlaan	142	3523 CJ	Utrecht	Grevelingen	2
266	Tine		Dhoore	Briljantlaan	142	3523 CJ	Utrecht	's Gravenhage	3
266	Tine		Dhoore	Briljantlaan	142	3523 CJ	Utrecht	Maarseveense	4
266	Tine		Dhoore	Briljantlaan	142	3523 CJ	Utrecht	Zevenhuizerpla	5
267	Kevin	van	Torre	Utrechtse Straatweg	16	3442 QW	De Meern	Oosterschelde	1
267	Kevin	van	Torre	Utrechtse Straatweg	16	3442 QW	De Meern	Grevelingen	2
267	Kevin	van	Torre	Utrechtse Straatweg	16	3442 QW	De Meern	's Gravenhage	3
267	Kevin	van	Torre	Utrechtse Straatweg	16	3442 QW	De Meern	Maarseveense	4
267	Kevin	van	Torre	Utrechtse Straatweg	16	3442 QW	De Meern	Zevenhuizerpla	5

Er klopt overduidelijk iets niet…

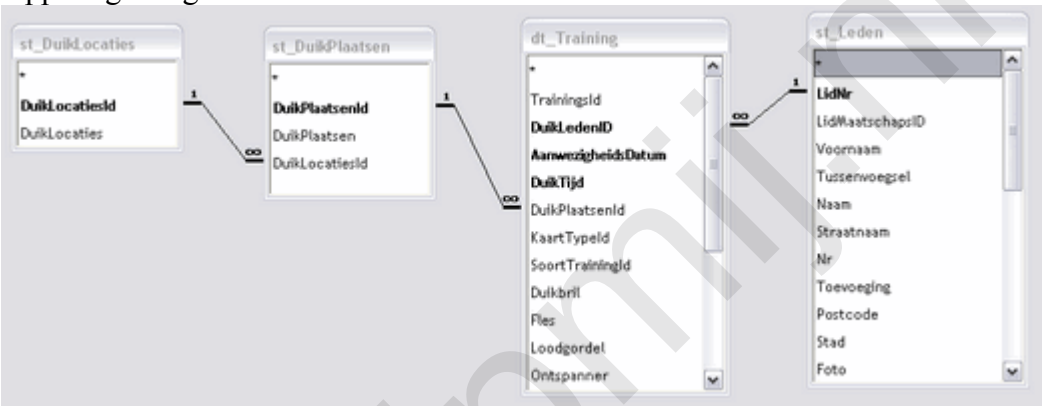
Het cartesisch product

We hebben een type query gemaakt die doorgaans niet erg bruikbaar is, zoals in dit geval: een *cartesisch product*. Als we het resultaat van de query ontleden, dan kunnen we vermoedelijk wel achterhalen wat er precies aan de hand is. Stel dat we in de tabel [Leden] 20 records hebben, en in de tabel [Locaties] 5 records. Als er 8 leden op de geselecteerde locatie hebben gedoken, dan zou je dus 8 records verwachten in de query. We zien er echter 100. En dat is exact het getal dat je krijgt als je 20 vermenigvuldigt met 5! Blijkbaar maakt Access voor elk lid 5 records aan met de unieke duiklocaties! Je ziet dat ook inderdaad terug in de query: de eerste 5 records zijn allemaal voor hetzelfde lid, alleen de locaties veranderen. Na 5 records zie je het volgend lid, en wederom de 5 verschillende duiklocaties. En zo gaat dat maar door.

Zoals gezegd, we noemen dat een cartesisch product. En soms kan dat handig zijn als je een lijst wil maken van alle combinaties die er mogelijk zijn. Maar niet hier dus!

Als je het Relaties venster opent, dan zul je zien, dat de tabel [Duiklocaties] indirect wel is gekoppeld aan de tabel [Leden]. Er zitten namelijk nog een paar tabellen tussen. De tabel [Duikleden] is eerst gekoppeld aan de tabel [Training], die op zijn beurt weer is gekoppeld aan de tabel [Duikplaatsen] en die is weer gekoppeld aan de tabel [Duiklocaties]. Oftewel: in de tabel [Training] leggen we vast welke leden op een training zijn geweest, en in welke plaats die training heeft plaatsgevonden. Dat gebeurt met een één-op-veel koppeling. De tabel [Plaats] heeft een één-op-veel koppeling met de tabel [Duiklocaties]. Of te wel: als we weten welke leden aan een training hebben meegedaan, weten we ook waar ze die training hebben gedaan, en als we weten in welke plaats ze zijn geweest, weten we ook op welke locatie!

Kortom: om in onze query de juiste gegevens te kunnen zien, moeten we alle noodzakelijke tabellen selecteren en toe voegen. Dus niet alleen de tabellen [Duikleden] en [duiklocaties], maar ook [Training] en [Duikplaatsen]. Als je deze tabellen toevoegt, zie je ook dat Access direct alle koppelingen legt tussen de tabellen.



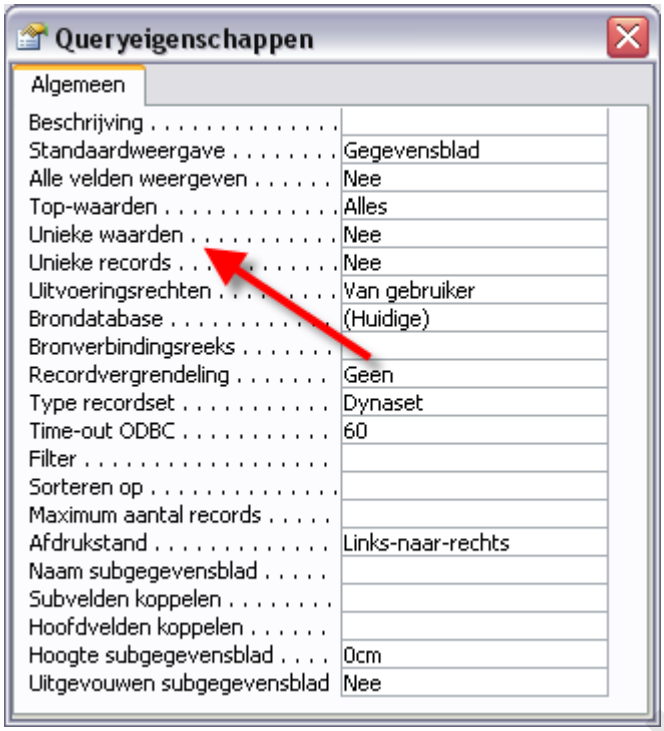
Als je de query nu uitvoert, heb je goede kans dat het aantal records zelfs nog groter is dan bij de variant met het Cartesisch product. Dat komt dan, doordat de tabel [Training] voor elke training van een duiklid een record opslaat. Het resultaat zal dus exact overeenkomen met het aantal records in de tabel [Training]. Dit levert op zich een bruikbaar resultaat op, maar onvermijdelijk zullen daar ook duplicaten tussen zitten: als een lid namelijk meer dan één training heeft gevolgd! Heeft iemand 5 trainingen gedaan, dan heeft deze persoon ook 5 records.

LidNr	Voornaam	Tussenvoegsel	Naam	Straatnaam	Nr	Postcode	Stad	DuikLocaties	DuikLocatiesId
280	Geert		Defever	Weerdsingel O.Z.	1	3514 AA	Utrecht	Oosterschelde	1
280	Geert		Defever	Weerdsingel O.Z.	1	3514 AA	Utrecht	Oosterschelde	1
270	Nick		Schiltz	Jan van Scorelstraat	158	3583 CV	Utrecht	Oosterschelde	1
270	Nick		Schiltz	Jan van Scorelstraat	158	3583 CV	Utrecht	Oosterschelde	1
282	Kristof		Deneire	Biltse Veste	25	3432 AP	Nieuwegein	Oosterschelde	1
283	Delphine	van	Damme	Bloemstraat	26	3581 WE	Utrecht	Oosterschelde	1
270	Nick		Schiltz	Jan van Scorelstraat	158	3583 CV	Utrecht	Oosterschelde	1

In het voorbeeld kun je zien dat Geert blijkbaar al twee keer in de Oosterschelde is geweest, Nick drie keer, en Kristof en Delphine één keer.

Een query filteren op unieke records

We moeten het resultaat van de query dus nog een beetje ‘finetunen’. We doen dat door een eigenschap van de query aan te passen. In onderstaande afbeelding zie je een aantal eigenschappen die je kunt instellen.



De eigenschap die we moeten aanpassen is in de afbeelding met een pijl aangegeven: de optie <Unieke waarden>. Die waarde veranderen van *Nee* naar **Ja**.

Als we de query nu opnieuw uitvoeren, krijgen we het gewenste resultaat:

LidNr	Voornaam	Tussenvoegsel	Naam	Straatnaam	Nr	Postcode	Stad	DuikLocaties	DuikLocatiesId
270	Nick		Schiltz	Jan van Scorelstraat	158	3583 CV	Utrecht	Oosterschelde	1
280	Geert		Defever	Weerdsingel O.Z.	1	3514 AA	Utrecht	Oosterschelde	1
282	Kristof		Deneire	Biltse Veste	25	3432 AP	Nieuwegein	Oosterschelde	1
283	Delphine	van	Damme	Bloemstraat	26	3581 WE	Utrecht	Oosterschelde	1

We zien alle personen één keer vermeld in het resultaat.

De functie Totalen

Er is nog een andere manier om unieke records te laten zien in een query. Daarvoor gebruiken we de knop *Totalen* () Als je deze knop aanzet, krijgt de query er een extra regel bij: *Totaal*. Je ziet ook, dat alle velden in de query standaard de functie *Group By* hebben gekregen. Deze functie zorgt er voor dat elk veld in beginsel eenmalig wordt weergegeven in de query. Als de query zodanig is gemaakt dat het resultaat ervan is dat de combinatie van alle velden uniek is, zul je elk record ook maar één keer zien terugkomen. Dit principe levert al geen unieke records meer op, als je bijvoorbeeld een veld als [Trainingsdatum] in de query zet. Probeer zelf te beredeneren waarom dat zo is…

Berekeningen maken in een query

Queries kun je gebruiken om (een deel van) een tabel te laten zien, en te filteren op de manier zoals hierboven is uitgelegd, maar je kunt in een query ook berekeningen maken. Denk daarbij bijvoorbeeld aan een tabel [Bestellingen], waarin je vastlegt wie welke goederen bestelt. Bij een artikel hoort uiteraard een prijs, en de prijs die de klant moet betalen is afhankelijk van het aantal exemplaren dat hij bestelt van een bepaald artikel.

Berekeningen op Recordniveau

In het hoofdstuk over *Normaliseren* heb je geleerd dat we gegevens die berekend kunnen worden doorgaans niet worden opgeslagen in een tabel. Netter gezegd: als we een bepaald gegeven, zoals

een totaalprijs, kunnen afleiden uit andere gegevens uit een tabel, dan slaan we dat gegeven niet op. In het voorbeeld van een besteld artikel geldt dat dus ook: in de tabel [Bestellingen] hebben we vastgelegd welk artikel de klant heeft besteld, en hoeveel exemplaren van dat artikel. De prijs die de klant moet betalen, is een eenvoudige formule: [TotaalPrijs] = [Aantal] * [Artikelprijs]

Zo'n berekening maken we dus in eerste instantie in een query. Bijvoorbeeld in een query die we gaan gebruiken om in een volgend hoofdstuk een factuur te maken. Die query is dan de basis voor de factuur.

Berekeningen op Groepsniveau

Als de knop *Totalen* is aangezet, kunnen we berekeningen maken op groepen gegevens. Bijvoorbeeld: hoe hoog is het totale bedrag voor een bepaalde bestelling? Aan de tabel [Bestelling] hangt de tabel [Bestelregels], die de afzonderlijke artikelen bevat. Door een query te maken, en te groeperen op het veld Bestelnummer, kunnen we met de functie *Som* berekenen wat het totaalbedrag is voor de bestelling.

Over berekeningen gaat een volgend hoofdstuk, dus daarom ga ik daar nu niet dieper op in.

Parameters in een query

Dat geldt ook voor het onderwerp *Parameters*. Ik wil al wel aanstippen wat dat inhoudt. Een parameter maakt een query niet alleen *dynamisch*, maar ook *interactief*. Een parameter kun je zien als een flexibel criteriumveld, waarin je waarden kunt typen waarop gefilterd moet worden. Je kunt een query bijvoorbeeld filteren op een begin- en einddatum met twee parametervelden voor een begindatum en een einddatum. Deze parameters kunnen zelfstandig zijn, zodat ze elke keer opnieuw moeten worden ingetypt, of afhankelijk worden gemaakt van een formulier. In het laatste geval gebeurt de filtering op een formulier, en worden de ingevulde waarden uit dat formulier ingelezen in de query.

Door parameters te gebruiken in een query, maak je de query dus uiterst flexibel. Vandaar dat ook dit onderwerp uitgebreid behandeld gaat worden in een volgend hoofdstuk.

Wanneer kan ik een query gebruiken om gegevens in te voeren?

Het laatste onderwerp dat ik nu wil behandelen is de vraag waarom een query de ene keer wel gebruikt kan worden om records in te voeren, en een volgende query dat niet doet.

Om die vraag te doorgronden, moeten we eerst vaststellen wat er gebeurt als we gegevens gaan invoeren. En dat is eigenlijk heel simpel: als ik een tabel open, kan ik records invoeren, verwijderen en muteren. Daarbij kan ik een record alleen opslaan als aan alle voorwaarden die aan die tabel hangen is voldaan. Dat houdt o.a. in dat alle verplichte velden zijn ingevuld, en dat aan de *Validatie-eisen* van de tabel is voldaan. Zo'n eis kan bijvoorbeeld zijn dat de eindtijd van een training nooit vóór de begintijd van de training kan liggen. Verder kan je maar in één tabel tegelijk mutaties verrichten, als de tabellen van elkaar afhankelijk zijn.

Datzelfde principe geldt ook voor een query: als je een query maakt om bestellingen te maken, omdat je in de query niet alleen de LeveranciersID wilt zien, maar ook zijn NAW gegevens, dan wil je met je query de tabel [Bestellingen] vullen. In die tabel is het veld [LeveranciersID] een verplicht veld. Je hebt, als je de tabel [Bestellingen] en de tabel [Leveranciers] hebt toegevoegd, in beide tabellen een veld [LeveranciersID]. Maak je een query die is bedoeld als overzichtsquery, dan maakt het niet uit welk van de twee velden [LeveranciersID] je gebruikt: ze zijn (logischerwijs; ga dat zelf na!) identiek.

Maar bij het maken van een Bestelling maakt het wel degelijk uit: zou je het veld [LeveranciersID] uit de tabel [Leveranciers] gebruiken, en verder alle velden uit [Bestellingen] in het Query-raster

zetten, dan zou je, bij het maken van een nieuw record, alle velden in [Bestellingen] invullen, *behalve* het veld [LeveranciersID]. En dat is nu juiste een verplicht veld in de tabel [Bestellingen]! En omdat je een verplicht veld niet invult, kan Access het record niet opslaan!

Er is nog een ander aspect aan deze constructie, die er voor zorgt dat het maken van een nieuw record niet lukt: niet alleen probeer je een nieuw record te maken in de tabel [Bestellingen], maar doordat je ook een veld uit de tabel [Leveranciers] hebt geselecteerd, probeer je ook een nieuw record te maken in de tabel [Leveranciers]. En in die tabel is het veld [LeveranciersID] een sleutelveld! En zoals je weet, mag je een sleutelveld in een tabel maar één keer gebruiken. Je kunt hetzelfde gegeven dus niet gebruiken om een nieuwe leverancier mee aan te maken. En eigenlijk wil je dat ook helemaal niet: je wilt in je bestelling een bestaande leverancier opslaan. Zelfs niet eens een onbekende leverancier, want dat mag (gezien de relaties) ook helemaal niet!

Kortom: als je een tabel wilt vullen of muteren met een query, dan moet je er voor zorgen dat alle velden in de in te vullen tabel worden gebruikt. Dat houdt voor een query voor Bestellingen in, dat je alle velden uit de tabel [Bestellingen] gebruikt, dus ook het veld [LeveranciersID]. Om de NAW gegevens van de leverancier te zien, kun je in je query wel alle Naam- en adres gegevens opnemen. Dit zijn namelijk geen sleutelgegevens, en dus vrij op te vragen.

Het resultaat van de query zal misschien (aangenaam) verrassend zijn: als je nu een nieuw record toevoegt, en in het veld [LeveranciersID] een (uiteraard bestaand) nummer intypt, zal Access alle overige NAW gegevens uit de tabel [Leveranciers] gelijk invullen! Deze automatische opzoekfunctie wordt getriggerd door de relatie tussen de tabellen [Leveranciers] en [Bestellingen]. Zodra er een Leveranciersnummer is ingevoerd, controleert Access of dit nummer ook bestaat in de tabel Leveranciers. En op basis van dit nummer kan Access de overige gevraagde gegevens, zoals de NAW gegevens, invullen in de query.

Met deze eigenschap kun je uiterst complete formulieren en rapporten (zoals het bestel- of factuur rapport) maken: je neemt alle gegevens die je wilt gebruiken op in je query, die is gebaseerd op het hoofdonderwerp, zoals de tabel [Trainingen] als je trainingsrecords wilt toevoegen, en Bestellingen als je een bestelling wilt maken. Access vult vervolgens alle opgevraagde velden uit de gekoppelde tabellen aan.

In een formulier is het op deze manier zelfs mogelijk om van een leverancier bijvoorbeeld de adresgegevens te veranderen! Zolang in een bestelling de [LeveranciersID] niet verandert, zal de query altijd de gegevens van de juiste leverancier opzoeken. De overige velden die je opvraagt, zijn, net als bij het openen van de tabel [Leveranciers] gewoon wijzigbaar. Verander je dus bijvoorbeeld het telefoonnummer van een leverancier, dan zul je zien dat in je query alle records van die leverancier gelijk worden aangepast! Dat is dus op zich een gevaarlijke eigenschap, die je op formulieren en rapporten dan ook zou moeten afschermen: je wilt immers niet dat een gebruiker per abuis bij het maken van een bestelling de NAW gegevens van de klant verandert!

Kortom: als een query is gebaseerd op één invoertabel, en er geen sleutelvelden uit andere tabellen in de query zijn opgenomen, dan kun je deze query gebruiken om records toe te voegen aan de gekozen invoertabel. Zit er daarentegen een sleutelveld uit een andere tabel in de query, of zitten niet alle verplichte velden uit de invoertabel in de query, dan zal het invoeren van records niet lukken.

Er zijn meer redenen waarom het invoeren van records m.b.v. een query niet lukt: de belangrijkste daarvan is wel dat de knop *Totalen* aanstaat, of de optie *Unieke waarden*. Dit onderwerp wordt later nog uitgebreider besproken.

Samenvatting

We hebben een begin gemaakt met het maken van queries. We hebben de verschillende typen queries benoemd, en ons vervolgens geconcentreerd op het belangrijkste type: de selectiequery. We

hebben een query gemaakt op basis van één tabel, en op basis van meerdere tabellen. Tevens hebben we gezien waarom de ene query wel kan worden gebruikt om gegevens in een tabel in te voeren, en een andere query niet. Het belang van queries in een database kan niet genoeg worden benadrukt: ze staan aan de basis van alle overzichten, worden gebruikt voor rapporten en formulieren, en zijn bijvoorbeeld ook de basis van Keuzelijsten. Kortom: queries komen we overal tegen in een database!

Volgende Aflevering

In de volgende aflevering gaan we dieper in op de werking van formulieren. We gaan een startscherm ontwerpen, waarin we verschillende formulieren met verschillende taken kunnen openen. We baseren een formulier op een query, en gaan keuzelijsten maken met queries.