

Databases ontwerpen

Beginselen van databaseontwerp

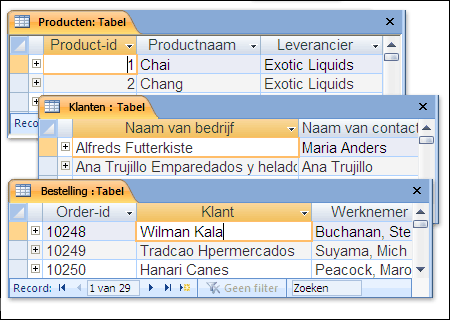
# Inleiding

Een correct ontworpen database biedt toegang tot accurate, bijgewerkte informatie. Aangezien een correct ontwerp van groot belang is om uw doel te bereiken bij het werken met een database, dient u voldoende tijd te nemen om de beginselen van een goed ontwerp te leren. Uiteindelijk beschikt u dan over een database die aan uw behoeften voldoet en die gemakkelijk kan worden gewijzigd.

In dit artikel worden richtlijnen gegeven voor het plannen van een database. U leert hoe u beslist welke gegevens u nodig hebt, hoe u die gegevens over de juiste tabellen en kolommen verdeelt en wat de relaties tussen die tabellen zijn. Lees dit artikel voordat u uw eerste database maakt.

# Enkele belangrijke databasetermen

Een database ordent uw gegevens in tabellen: lijsten met rijen en kolommen die doen denken aan boekhoudpapier of een spreadsheet. Een eenvoudige database kan uit slechts één tabel bestaan. Voor de meeste databases hebt u echter meer dan één tabel nodig. U kunt bijvoorbeeld een tabel maken met gegevens over producten, een andere tabel met gegevens over orders en nog een tabel met gegevens over klanten.



Elke rij wordt ook wel een record genoemd, terwijl elke kolom ook wel een veld wordt genoemd. Een record vormt een betekenisvolle en consistente manier om informatie over iets te combineren. Een veld bestaat uit één item met informatie, een itemtype dat in elke record voorkomt. In de tabel Producten kan elke rij of record bijvoorbeeld gegevens bevatten over één product. Elke kolom of veld bevat dan een bepaald type gegevens over dat product, zoals de naam of de prijs.

Rij: record -

Kolom: veld |

# Wat is een goed databaseontwerp?

Er gelden bepaalde basisprincipes voor databaseontwerp. Het eerste principe is dat **dubbele gegevens (ook wel redundante gegevens genoemd) niet mogen voorkomen**, omdat hierdoor ruimte wordt verspild en de kans op fouten en inconsistenties toeneemt. Het tweede principe is dat de **gegevens correct en volledig** moeten zijn. Als uw database onjuiste gegevens bevat, bevatten rapporten die informatie uit die database halen, ook onjuiste gegevens. Daardoor zijn beslissingen die u neemt op basis van die rapporten op onjuiste gegevens gebaseerd.

Een goed databaseontwerp is daarom een ontwerp dat:

* Uw gegevens opsplitst in tabellen op basis van onderwerp om zo redundante gegevens te voorkomen.
* Uw database de gegevens levert die het programma nodig heeft om de informatie in de tabellen op de gewenste manier samen te voegen.
* Helpt de nauwkeurigheid en integriteit van de gegevens te garanderen.
* Voldoet aan uw wensen op het gebied van gegevensverwerking en -rapportage.

# Het ontwerpproces

Het ontwerpproces bestaat uit de volgende stappen:

1. Het doel van de database bepalen      
   Dit helpt u bij de voorbereiding op de overige stappen.
2. De vereiste gegevens zoeken en organiseren      
   Verzamel alle soorten informatie die u mogelijk wilt vastleggen in de database, zoals productnaam en ordernummer.
3. De gegevens opsplitsen in tabellen      
   Splits de gegevensitems op in hoofdentiteiten of onderwerpen, zoals Producten of Orders. Elk onderwerp wordt dan een tabel.
4. Gegevensitems omzetten in kolommen      
   Bepaal welke gegevens u in elke tabel wilt opslaan. Elk item wordt een veld en wordt weergegeven als kolom in de tabel. Een tabel met werknemers kan bijvoorbeeld velden zoals Achternaam en Datum in dienst bevatten.
5. Primaire sleutels opgeven      
   Kies voor elke tabel een primaire sleutel. De primaire sleutel is een kolom die wordt gebruikt om elke rij op unieke wijze te identificeren. Een voorbeeld hiervan is Product-id of Ordernummer.
6. Tabelrelaties instellen      
   Bekijk elke tabel en bepaal de relatie tussen de gegevens in de ene tabel en de gegevens in andere tabellen. Voeg zo nodig velden toe aan tabellen of maak nieuwe tabellen om de relaties duidelijker vast te leggen.
7. Het ontwerp bijschaven      
   Analyseer uw ontwerp op fouten. Maak de tabellen en voeg enkele records met voorbeeldgegevens toe. Kijk of de tabellen het gewenste resultaat opleveren. Pas het ontwerp zo nodig aan.
8. De normalisatieregels toepassen      
   Pas de normalisatieregels voor gegevens toe om te kijken of de structuur van uw tabellen correct is. Pas de tabellen zo nodig aan.

# Het doel van de database bepalen

Het is aan te raden het doel van de database op papier te noteren. Noteer hierbij het doel, de manier waarop u de database verwacht te gebruiken en wie deze gaat gebruiken. Voor een kleine database voor een bedrijf aan huis kunt u bijvoorbeeld een eenvoudige beschrijving gebruiken, bijvoorbeeld: 'In de klantendatabase wordt een lijst bijgehouden met klantgegevens die worden gebruikt om mailings en rapporten te maken.' Als de database complexer is of door veel mensen wordt gebruikt, wat vaak het geval is in een groot bedrijf, kan het doel uit een of meer alinea's bestaan waarin wordt aangegeven wanneer en hoe elke persoon de database gebruikt. Het komt erop neer dat u over een goed uitgewerkte doelstelling moet beschikken die tijdens het ontwerpproces kan worden geraadpleegd. Een dergelijke doelstelling helpt u uw aandacht te richten op uw doelen wanneer u beslissingen neemt.

# De vereiste gegevens zoeken en ordenen

Wanneer u de vereiste gegevens zoekt en ordent, begint u met de bestaande gegevens. Het is bijvoorbeeld mogelijk dat u inkooporders bijhoudt in een grootboek of klantgegevens bijhoudt op papieren formulieren in een archiefkast. Verzamel die documenten en noteer elk type informatie die zij bevatten (bijvoorbeeld elk vakje dat u invult op een formulier). Als u geen bestaande formulieren hebt, doe dan alsof u een formulier moet ontwerpen waarop de klantgegevens worden ingevuld. Welke gegevens zou u op dat formulier zetten? Welke invoervakken zou u maken? Bepaal en noteer elk van deze items. Stel, u houdt momenteel een klantenlijst bij op indexkaarten. Deze kaarten bevatten waarschijnlijk een klantnaam, adres, postcode, plaats en telefoonnummer. Elk van deze items vertegenwoordigt een mogelijke kolom in een tabel.

Wanneer u deze lijst voorbereidt, hoeft deze niet direct perfect te zijn. Noteer in plaats daarvan elk item dat u te binnen schiet. Als iemand anders de database gebruikt, vraag die persoon dan eveneens om ideeën. U kunt de lijst later verder bijschaven.

Kijk vervolgens welke soorten rapporten of mailings u wilt maken op basis van de database. U wilt bijvoorbeeld een rapport maken over de productverkoop per regio of een voorraadoverzicht waarin de voorraad van elk product wordt aangegeven. U wilt misschien ook standaardbrieven voor klanten genereren waarin een verkoopevenement wordt aangekondigd of een aanbieding wordt gedaan. Ontwerp het rapport in gedachten en stel u voor hoe dit eruit zal zien. Bedenk welke gegevens u in het rapport zou willen opnemen. Noteer elk item. Doe hetzelfde voor de standaardbrief en voor elk ander rapport dat u verwacht te maken.



Door na te denken over de rapporten en mailings die u wilt maken, kunt u beter bepalen welke items u nodig hebt in de database. Stel dat uw klanten zich kunnen aanmelden (of afmelden) voor een periodieke nieuwsbrief die per e-mail wordt verstuurd en dat u een lijst wilt afdrukken met klanten die zich hebben aangemeld. Om die informatie vast te leggen, voegt u de kolom E-mail verzenden toe aan de klantentabel. Voor elke klant kunt u het veld dan instellen op Ja of Nee.

Als u e-mailberichten naar klanten wilt kunnen sturen, moet u nog een item opslaan. Als u eenmaal weet of een klant e-mail wil ontvangen, moet u immers ook het e-mailadres kennen waar u die berichten naartoe moet sturen. Daarom moet u een e-mailadres voor elke klant kunnen opslaan.

Het is aan te raden een prototype van elk rapport en elke uitvoerlijst te maken en te kijken welke items u nodig hebt om het rapport te genereren. Als u bijvoorbeeld een standaardbrief wilt genereren, hebt u bepaalde gegevens nodig. Als u de juiste titel voor de aanhef wilt toevoegen, zoals heer of mevrouw, moet u een item voor de titel maken. U begint een brief waarschijnlijk ook meestal met iets als 'Beste meneer Wolthuis', in plaats van 'Beste Dhr. Wander Wolthuis'. Dit betekent dat u de voornaam en de achternaam gewoonlijk apart opslaat.

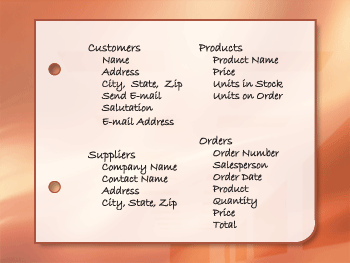
Een belangrijk punt dat u hierbij moet onthouden, is dat u elk stukje informatie moet opsplitsen in de kleinste nuttige delen. Als u in het geval van een naam de achternaam afzonderlijk wilt kunnen gebruiken, moet u de naam in tweeën delen: Voornaam en Achternaam. Als u een rapport bijvoorbeeld wilt sorteren op achternaam, dient u de achternaam van de klanten afzonderlijk op te slaan. In het algemeen kunt u het beste een apart veld gebruiken voor gegevens die u moet kunnen sorteren, waarin u moet kunnen zoeken, die in berekeningen worden gebruikt of waarover moet worden gerapporteerd.

Denk na over de vragen die de database moet beantwoorden. Bijvoorbeeld: hoeveel exemplaren van een bepaald product zijn de afgelopen maand verkocht? Waar wonen uw beste klanten? Wie is de leverancier van uw bestverkochte product? Door vooruit te lopen op dergelijke vragen, kunt u beter bepalen welke aanvullende items u moet vastleggen.

Nadat u deze informatie hebt verzameld, kunt u verdergaan met de volgende stap.

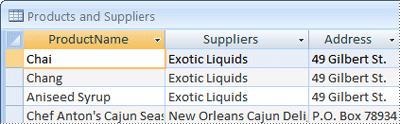
# De gegevens opsplitsen in tabellen

U splitst de gegevens op in tabellen door de belangrijkste entiteiten of onderwerpen te kiezen. Nadat u de gegevens voor een database voor de productverkoop hebt gevonden en geordend, kan de voorlopige lijst er als volgt uitzien:



De hoofdentiteiten die hier worden weergegeven, zijn producten, leveranciers, klanten en orders. Daarom is het zinvol om met de volgende vier tabellen te beginnen: een tabel met gegevens over producten, een met gegevens over leveranciers, een met gegevens over klanten en een met gegevens over orders. Hoewel de lijst hiermee niet klaar is, vormt dit een goed uitgangspunt. U kunt deze lijst later verder bijschaven totdat u over een geschikt ontwerp beschikt.

Wanneer u de voorlopige lijst met items voor het eerst controleert, bent u misschien geneigd alle items in één tabel te plaatsen in plaats van in de vier tabellen uit de voorgaande illustratie. Hier wordt uitgelegd waarom dat geen goed idee is. Kijk eens naar deze tabel:



In dit geval bevat elke rij informatie over zowel het product als de leverancier. Aangezien u veel producten bij één leverancier kunt betrekken, moeten de naam- en adresgegevens van de leverancier vele malen worden herhaald. Dit is een verspilling van schijfruimte. Het is een betere oplossing om de gegevens van de leverancier slechts eenmaal op te slaan in een aparte tabel Leveranciers en die tabel vervolgens te koppelen aan de tabel Producten.

Een tweede probleem met dit ontwerp doet zich voor wanneer u de gegevens over de leverancier moet wijzigen. Stel, u moet het adres van een leverancier wijzigen. Aangezien dit adres op veel plaatsen wordt gebruikt, zou u het adres per ongeluk op de ene plaats kunnen wijzigen, terwijl u vergeet het op andere plekken te wijzigen. Dit probleem wordt opgelost als u het adres van de leverancier op slechts één plek opslaat.

Wanneer u een database ontwerpt, moet u altijd proberen ervoor te zorgen dat elk gegeven slechts eenmaal wordt opgeslagen. Als u merkt dat u dezelfde informatie, zoals het adres van een leverancier, op meer dan één plek moet invoeren, plaatst u die informatie in een aparte tabel.

Stel tot slot dat u slechts één product betrekt bij Coho Winery en dat u dat product wilt verwijderen, maar de naam- en adresgegevens van de leverancier wilt bewaren. Hoe kunt u dan de productrecord verwijderen zonder ook de leveranciersgegevens kwijt te raken? Dat is niet mogelijk. Aangezien elke record informatie bevat over een product en informatie over een leverancier, kunt u het een niet verwijderen zonder het ander eveneens te verwijderen. Als u deze gegevens afzonderlijk wilt bewaren, moet u de tabel in tweeën splitsen: één tabel voor productgegevens en een andere tabel voor leveranciersgegevens. Als u in dat geval een productrecord verwijdert, worden alleen de gegevens over het product verwijderd, niet de gegevens over de leverancier.

Nadat u het ontwerp voor een tabel hebt bepaald, slaat u in de kolommen in die tabel alleen gegevens over dat onderwerp op. In de producttabel slaat u bijvoorbeeld alleen gegevens over producten op. Aangezien het adres van de leverancier informatie over de leverancier vormt en niet over het product, behoort dit tot de leverancierstabel.

# Gegevensitems omzetten in kolommen

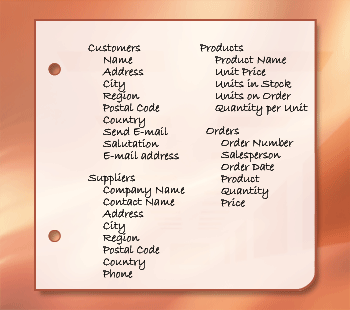
Als u wilt bepalen welke kolommen een tabel moet bevatten, stelt u vast welke gegevens u wilt bijhouden over het onderwerp dat wordt vastgelegd in de tabel. Voor de tabel Klanten vormen Naam, Adres, Postcode-Plaats, E-mail verzenden, Titel en E-mailadres een goed uitgangspunt voor de lijst met kolommen. Elke record in de tabel bevat dezelfde reeks kolommen, zodat u gegevens over naam, adres, postcode en plaats, e-mail verzenden, titel en e-mailadres voor elke record kunt opslaan. De kolom Adres bevat bijvoorbeeld het adres van de klanten. Elke record bevat de gegevens over één klant, terwijl het adresveld het adres van die klant bevat.

Nadat u de aanvankelijke reeks kolommen voor elke tabel hebt bepaald, kunt u de kolommen verder uitwerken. Het is bijvoorbeeld zinvol om de naam van de klant op te slaan in twee aparte kolommen, Voornaam en Achternaam, zodat u de gegevens kunt sorteren, doorzoeken en indexeren op alleen die kolommen. Op dezelfde manier bestaat het adres uit vijf verschillende onderdelen, namelijk adres, postcode, plaats, provincie en land/regio, zodat het zinvol is die onderdelen eveneens in aparte kolommen op te slaan. Als u bijvoorbeeld wilt zoeken, filteren of sorteren op provincie, moet u de provincie opslaan in een aparte kolom.

U moet ook bepalen of de database alleen Nederlandse gegevens of ook internationale informatie bevat. Als u bijvoorbeeld internationale adressen wilt opslaan, kunt u beter een kolom Regio dan een kolom Provincie gebruiken, aangezien deze kolom zowel Nederlandse provincies als de regio's voor andere landen kan bevatten.

Hieronder worden enkele tips gegeven voor het bepalen van de kolommen.

* **Gebruik geen berekende gegevens**Meestal kunt u het resultaat van berekeningen beter niet in tabellen opslaan. In plaats daarvan kunt u de berekeningen laten uitvoeren wanneer u het resultaat wilt zien. Stel, in het rapport Producten in bestelling wordt het subtotaal van het aantal bestelde artikelen voor elke productcategorie in de database aangegeven. Geen enkele tabel bevat echter de subtotaalkolom Eenheden in bestelling. In plaats daarvan bevat de tabel Producten een kolom Eenheden in bestelling waarin het aantal bestelde eenheden voor elk product wordt opgeslagen. Telkens wanneer u het rapport afdrukt, wordt het subtotaal berekent op basis van die gegevens. Het subtotaal zelf mag niet in een tabel worden opgeslagen.
* **Sla informatie op in de kleinst mogelijke delen**U bent misschien geneigd één veld te maken voor de voor- en achternaam of voor productnamen met de productbeschrijving. Als u meer dan één soort informatie combineert in een veld, kunt u de afzonderlijke gegevens moeilijk achteraf ophalen. Probeer gegevens op te splitsten in logische delen. Maak bijvoorbeeld aparte velden voor voor- en achternaam, of voor productnaam, categorie en beschrijving.



Nadat u de gegevenskolommen in elke tabel hebt uitgewerkt, kunt u voor elke tabel een primaire sleutel gaan kiezen.

# Primaire sleutels opgeven

Elke tabel moet een kolom of reeks kolommen bevatten die elke rij in de tabel op unieke wijze identificeren. Dit is vaak een uniek identificatienummer, zoals een werknemer-id of een serienummer. In databaseterminologie wordt deze informatie de primaire sleutel van de tabel genoemd. In een database worden primaire-sleutelvelden gebruikt om gegevens uit meerdere tabellen snel te koppelen en de gegevens voor u samen te voegen.

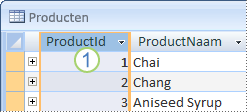
Als een tabel al een unieke id bevat, zoals een productnummer dat elk product in de catalogus op unieke wijze identificeert, kunt u deze id als primaire sleutel voor de tabel gebruiken, maar alleen als de waarden in die kolom altijd voor elke record verschillend zijn. Een primaire sleutel mag geen dubbele waarden bevatten. U kunt namen van personen bijvoorbeeld niet als primaire sleutel gebruiken, omdat namen niet uniek zijn. Een tabel kan immers twee personen met dezelfde naam bevatten.

Een primaire sleutel moet altijd een waarde hebben. Als de waarde van een kolom op enig moment wordt verwijderd of onbekend wordt (een ontbrekende waarde), kan deze niet als primaire sleutel worden gebruikt.

U dient altijd een primaire sleutel te kiezen waarvan de waarde niet verandert. In een database met meer dan één tabel, kan de primaire sleutel van een tabel worden gebruikt als verwijzing in andere tabellen. Als de primaire sleutel verandert, moet deze wijziging ook overal worden toegepast waar naar de sleutel wordt verwezen. Als u een primaire sleutel gebruikt die niet verandert, verkleint u de kans dat de primaire sleutel niet meer synchroon is met andere tabellen die ernaar verwijzen.

Vaak wordt een willekeurig uniek nummer als primaire sleutel gebruikt. U kunt bijvoorbeeld aan elke order een uniek ordernummer toewijzen. Het enige doel van het ordernummer is een order te identificeren. Nadat het nummer is toegewezen, verandert dit nooit meer.

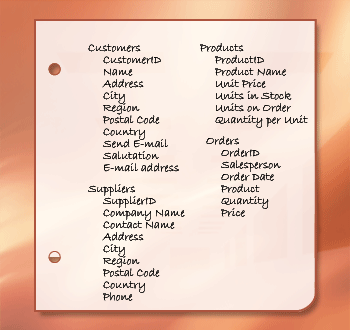
Als u geen kolom of reeks kolommen in gedachten hebt die een goede primaire sleutel kunnen vormen, overweeg dan een kolom met het gegevenstype AutoNummering te gebruiken. Wanneer u het gegevenstype AutoNummering gebruikt, wordt automatisch een waarde toegewezen. Dergelijke id's bevatten geen feitelijke gegevens die de rij beschrijven. Ze zijn ideaal voor gebruik als primaire sleutel omdat ze niet veranderen. De kans dat een primaire sleutel met gegevens over een rij, zoals een telefoonnummer of een klantnaam, verandert, is groter omdat de gegevens zelf kunnen veranderen.



1. Een kolom die is ingesteld op het gegevenstype AutoNummering vormt een goede primaire sleutel. Twee product-id's zijn namelijk nooit gelijk.

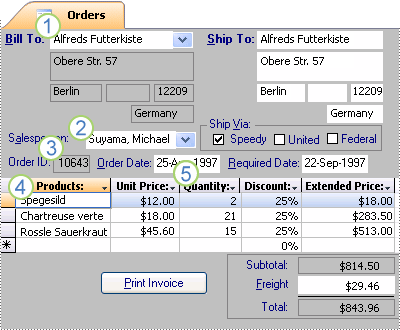
Soms kan het handig zijn om twee of meer velden te gebruiken die samen de primaire sleutel van een tabel vormen. Voor de tabel Ordergegevens, waarin artikelen voor orders worden opgeslagen, worden bijvoorbeeld twee kolommen gebruikt als primaire sleutel: Ordernummer en Product-id. Een primaire sleutel die gebruikmaakt van meer dan één kolom wordt ook wel een samengestelde sleutel genoemd.

Voor de database voor de productverkoop kunt u een AutoNummering-kolom maken als primaire sleutel voor elke tabel: Product-id voor de tabel Producten, Order-id voor de tabel Orders, Klant-id voor de tabel Klanten en Leverancier-id voor de tabel Leveranciers.



# Tabelrelaties maken

Nadat u de gegevens hebt opgesplitst in tabellen, hebt u een manier nodig om die gegevens op zinvolle manieren samen te voegen. Het volgende formulier bevat bijvoorbeeld gegevens uit verschillende tabellen.



1. De gegevens in dit formulier zijn afkomstig uit de tabel Klanten...

2. ...de tabel Werknemers...

3. ...de tabel Orders...

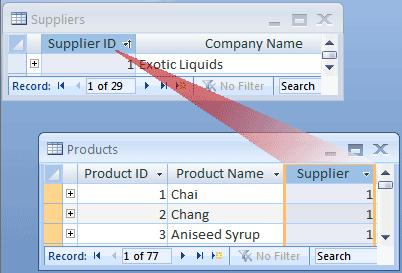
4. ...de tabel Producten...

5. ...en de tabel Ordergegevens.

SQL Server is een relationeel databasebeheersysteem. In een relationele database verdeelt u gegevens over aparte tabellen op basis van onderwerp. Vervolgens gebruikt u de tabelrelaties om de gegevens naar wens samen te voegen.

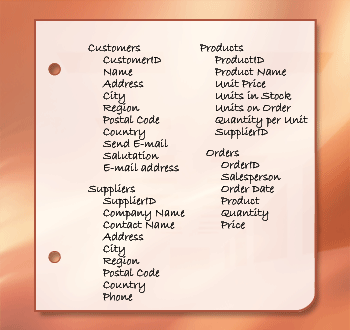
# Een één-op-veel-relatie maken

Kijk eens naar het volgende voorbeeld: de tabellen Leveranciers en Producten in de database met productorders. Een leverancier kan elk aantal producten leveren. Dit betekent dat de tabel Producten voor elke leverancier in de tabel Leveranciers veel producten kan bevatten. De relatie tussen de tabel Leveranciers en de tabel Producten is daarom een één-op-veel-relatie.



Als u een één-op-veel-relatie wilt weergeven in uw databaseontwerp, neemt u de primaire sleutel van de 'een'-kant van de relatie en voegt u deze als extra kolom(men) toe aan de tabel aan de 'veel'-kant van de relatie. In dit geval voegt u bijvoorbeeld de kolom Leverancier-id uit de tabel Leveranciers toe aan de tabel Producten. De database kan de leverancier-id in de tabel Producten vervolgens gebruiken om de juiste leverancier voor elk product te vinden.

De kolom Leverancier-id in de tabel Producten wordt een refererende sleutel genoemd. Een refererende sleutel is de primaire sleutel van een andere tabel. De kolom Leverancier-id in de tabel Producten is een refererende sleutel omdat deze tevens de primaire sleutel van de tabel Leveranciers is.



Verwante tabellen kunnen worden samengevoegd op basis van koppelingen tussen primaire sleutels en refererende sleutels. Als u niet zeker weet welke tabellen dezelfde kolom moeten gebruiken, kunt u een één-op-veel-relatie instellen. Hiervoor moeten de twee tabellen wel een gemeenschappelijke kolom hebben.

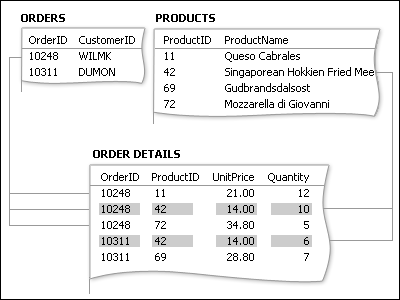
# Een veel-op-veel-relatie maken

Kijk eens naar de relatie tussen de tabel Producten en de tabel Orders.

Eén order kan meer dan één product bevatten. Anderzijds kan één product voorkomen in veel orders. Daarom kan de tabel Producten veel records bevatten voor elke record in de tabel Orders. En voor elke record in de tabel Producten kan de tabel Orders veel records bevatten. Dit typt relatie wordt een veel-op-veel-relatie genoemd omdat er voor elk product vele orders kunnen zijn en voor elke order vele producten. Als u veel-op-veel-relaties tussen tabellen wilt opsporen, moet u letten op beide zijden van de relatie.

De onderwerpen van de twee tabellen, orders en producten, hebben een veel-op-veel-relatie. Dit leidt echter tot een probleem. Om dit probleem te begrijpen, kijken we wat er zou gebeuren als u de relatie tussen de twee tabellen probeert te maken door het veld Product-id toe te voegen aan de tabel Orders. Als u meer dan één product per order wilt, moet de tabel Orders meer dan één record per order bevatten. U zou de ordergegevens moeten herhalen voor elke rij die betrekking heeft op één order, wat resulteert in een inefficiënt ontwerp, dat tot onjuiste gegevens kan leiden. U ondervindt hetzelfde probleem als u het veld Ordernummer in de tabel Producten plaatst: de tabel Producten zou dan meer dan één record voor elk product bevatten Hoe lost u dit probleem op?

Het antwoord hierop is een derde tabel te maken, ook wel een verbindingstabel genoemd, die de veel-op-veel-relatie splitst in twee één-op-veel-relaties. U voegt de primaire sleutel uit elk van de twee tabellen toe aan de derde tabel. Op die manier wordt in de derde tabel elke instantie van de relatie vastgelegd.



Elke record in de tabel Ordergegevens vertegenwoordigt één artikel in een order. De primaire sleutel van de tabel Ordergegevens bestaat uit twee velden: de refererende sleutels uit de tabellen Orders en Producten. U kunt niet alleen het veld Ordernummer als primaire sleutel voor deze tabel gebruiken, omdat één order meerdere artikelen kan bevatten. Het ordernummer wordt herhaald voor elk artikel in een order, zodat dit veld geen unieke waarden bevat. U kunt ook niet alleen het veld Product-id gebruiken, omdat één product kan voorkomen in meerdere orders. Samen leveren de twee velden echter een unieke waarde op voor elke record.

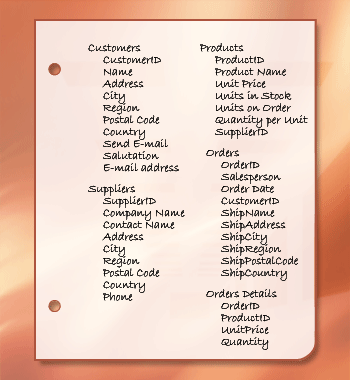
In de database voor de productverkoop zijn de tabellen Orders en Producten niet rechtstreeks aan elkaar gekoppeld. In plaats daarvan zijn ze indirect gekoppeld via de tabel Ordergegevens. De veel-op-veel-relatie tussen orders en producten wordt in de database gerepresenteerd door twee één-op-veel-relaties:

Er bestaat een één-op-veel-relatie tussen de tabel Orders en de tabel Ordergegevens. Elke order kan meer dan één artikel bevatten, maar elk artikel is gekoppeld aan slechts een order.

Er bestaat een één-op-veel-relatie tussen de tabel Producten en de tabel Ordergegevens. Aan elk product kunnen meerdere artikelen zijn gekoppeld, maar elk artikel verwijst slechts naar een product.

Vanuit de tabel Ordergegevens kunt u alle producten voor een bepaalde order vaststellen. U kunt ook alle orders voor een bepaald product bepalen.

Nadat u de tabel Ordergegevens hebt gemaakt, kan de lijst met tabellen en velden er bijvoorbeeld als volgt uitzien:



# Een een-op-een-relatie maken

Een ander type relatie is de één-op-één-relatie. Stel, u wilt speciale aanvullende productgegevens opslaan die u zelden nodig hebt of die alleen van toepassing zijn op enkele producten. Aangezien u deze gegevens niet vaak nodig hebt en aangezien de opslag van deze gegevens in de tabel Producten zou leiden tot lege ruimte voor elk product waarop de gegevens niet van toepassing zijn, plaatst u ze in een aparte tabel. Net als in de tabel Producten gebruikt u Product-id als primaire sleutel. De relatie tussen deze aanvullende tabel en de tabel Producten is een één-op-één-relatie. Voor elke record in de tabel Producten bestaat er één bijbehorende record in de aanvullende tabel. Wanneer u een dergelijke relatie creëert, moeten beide tabellen een gemeenschappelijk veld bevatten.

Wanneer u een één-op-één-relatie nodig hebt in een database, kijkt u of u de gegevens uit de twee tabellen samen in één tabel kunt plaatsen. Als u dat om de een of andere reden niet wilt doen, bijvoorbeeld omdat dit zou leiden tot veel lege velden, kunt u in de volgende lijst zien hoe u de relatie in uw ontwerp kunt representeren:

Als de twee tabellen hetzelfde onderwerp hebben, kunt u de relatie waarschijnlijk instellen door dezelfde primaire sleutel in beide tabellen te gebruiken.

Als de twee tabellen verschillende onderwerpen met verschillende primaire sleutels hebben, plaatst u de primaire sleutel van een van de tabellen als refererende sleutel in de andere tabel.

Door de relaties tussen tabellen te bepalen, zorgt u dat u over de juiste tabellen en kolommen beschikt. Als er een één-op-één- of één-op-veel-relatie bestaat, moeten de desbetreffende tabellen een of meer gemeenschappelijke kolommen hebben. Als er een veel-op-veel-relatie bestaat, is een derde tabel nodig die deze relatie vertegenwoordigt.

# Het ontwerp bijschaven

Nadat u de benodigde tabellen, velden en relaties hebt bepaald, moet u de tabellen maken en vullen met voorbeeldgegevens en met de gegevens proberen te werken door query's te maken, nieuwe records toe te voegen en dergelijke. Dit helpt u mogelijke problemen op te sporen. Het is bijvoorbeeld mogelijk dat u een kolom moet toevoegen die u tijdens de ontwerpfase bent vergeten of dat u een tabel in twee tabellen moet opsplitsen om dubbele gegevens te voorkomen.

Kijk of u de database kunt gebruiken om de benodigde antwoorden te krijgen. Maak een voorlopige versie van formulieren en rapporten, en controleer of daarin de verwachte gegevens worden weergegeven. Let op onnodige dubbele gegevens en pas in dat geval het ontwerp aan om dit te voorkomen.

Terwijl u de aanvankelijke database uitprobeert, ontdekt u waarschijnlijk dat er ruimte voor verbetering is. Hier volgen enkele zaken waarop u moet letten:

* Bent u kolommen vergeten? Zo ja, horen de gegevens thuis in de bestaande tabellen? Als de gegevens betrekking hebben op iets anders, moet u mogelijk een nieuwe tabel maken. Maak een kolom voor elk gegevensitem dat u wilt bijhouden. Als de gegevens niet kunnen worden berekend op basis van andere kolommen, moet u er waarschijnlijk een nieuwe kolom voor maken.
* Zijn bepaalde kolommen overbodig doordat ze kunnen worden berekend op basis van bestaande velden? Als een gegevensitem kan worden berekend op basis van bestaande kolommen, zoals een kortingsprijs die wordt berekend op basis van de winkelprijs, kunt u de gegevens beter berekenen en geen nieuwe kolom maken.
* Voert u herhaaldelijk dezelfde gegevens in een van de tabellen in? Zo ja, dan moet u de tabel waarschijnlijk opsplitsen in twee tabellen met een één-op-veel-relatie.
* Zijn er tabellen met veel velden, een beperkt aantal records en veel lege velden in afzonderlijke records? Zo ja, overweeg dan het tabelontwerp aan te passen, zodat de tabel minder velden en meer records bevat.
* Is elk gegevensitem opgesplitst in de kleinst mogelijke bruikbare delen? Plaats gegevens die u moet kunnen sorteren, waarin u moet kunnen zoeken, die in berekeningen worden gebruikt of waarover moet worden gerapporteerd in een eigen kolom.
* Bevat elke kolom gegevens over het onderwerp van de tabel? Als een kolom geen gegevens bevat over het onderwerp van de tabel, hoort de kolom in een andere tabel.
* Worden alle relaties tussen tabellen vertegenwoordigd door gemeenschappelijke velden of door een derde tabel? Eén-op-één- en één-op-veel-relaties vereisen gemeenschappelijke kolommen. Veel-op-veel-relaties vereisen een derde tabel.

## De tabel Producten bijschaven

Stel, elk product in de database voor de productverkoop valt binnen een algemene categorie, zoals dranken, kruiden of vis. De tabel Producten kan dan een veld bevatten met de categorie van elk product.

Nadat u het ontwerp van de database hebt gecontroleerd en aangepast, besluit u een beschrijving van de categorie bij de naam op te slaan. Als u het veld Categoriebeschrijving toevoegt aan de tabel Producten, moet u elke categoriebeschrijving herhalen voor elk product in die categorie. Dat is echter geen goede oplossing.

Een betere oplossing is om van Categorieën een nieuw onderwerp voor de database te maken, met een eigen tabel en een eigen primaire sleutel. U kunt de primaire sleutel uit de tabel Categorieën dan als refererende sleutel toevoegen aan de tabel Producten.

De tabellen Categorieën en Producten hebben een één-op-veel-relatie: een categorie kan meer dan één product bevatten, maar een product kan slechts tot één categorie behoren.

Wanneer u de structuur van de tabellen controleert, let dan op groepen gegevens die worden herhaald. Kijk bijvoorbeeld eens naar de tabel met de volgende kolommen:

* Product-id
* Naam
* Product-id1
* Naam1
* Product-id2
* Naam2
* Product-id3
* Naam3

Hier bestaat elk product uit een herhaalde groep kolommen die alleen van de andere groepen verschilt door het nummer aan het einde van de kolomnaam. Als u dergelijke genummerde kolommen ziet, dient u het ontwerp te herzien.

Zo'n ontwerp brengt diverse problemen met zich mee. Ten eerste moet u een bovengrens voor het aantal producten instellen. Zodra die grens wordt overschreden, moet u een nieuwe groep kolommen toevoegen aan de tabelstructuur, hetgeen een zware administratieve taak vormt.

Een ander probleem is dat voor leveranciers met minder dan het maximum aantal producten ruimte wordt verspild, aangezien de aanvullende kolommen leeg zijn. Het grootste probleem met een dergelijk ontwerp is echter dat veel taken moeilijk kunnen worden uitgevoerd, zoals de tabel sorteren of indexeren op product-id of naam.

Wanneer u herhaalde groepen tegenkomt, moet u het ontwerp nogmaals goed overwegen en kijken of u de tabel in tweeën kunt splitsen. In het bovenstaande voorbeeld kunt u beter twee tabellen gebruiken, een voor leveranciers en een voor producten, gekoppeld via de leverancier-id.

# De normalisatieregels toepassen

Als volgende stap in het ontwerp kunt u de normalisatieregels voor gegevens (soms eenvoudig normalisatieregels genoemd) toepassen. U gebruikt deze regels om te kijken of de structuur van de tabellen correct is. Het toepassen van deze regels op het databaseontwerp wordt normaliseren of normalisatie genoemd.

Normalisatie is het meest zinvol als alle gegevensitems zijn vertegenwoordigd en de voorlopige versie van het ontwerp is voltooid. Deze regels helpen u ervoor te zorgen dat u de gegevensitems in de juiste tabellen hebt gesplitst. Normalisatie kan er echter niet voor zorgen dat u over alle juiste gegevensitems beschikt.

U past de regels na elkaar toe, waarbij u er in elke stap voor zorgt dat het ontwerp een van de zogeheten 'normaalvormen' bereikt. Er bestaan vijf algemeen geaccepteerde normaalvormen: de eerste normaalvorm tot en met de vijfde normaalvorm. In dit artikel wordt nader ingegaan op de eerst drie vormen, omdat alleen deze drie zijn vereist voor de meeste databaseontwerpen.

## Eerste normaalvorm

De eerste normaalvorm schrijft voor dat er op elk snijpunt van een rij en een kolom in de tabel slechts één waarde bestaat en nooit een lijst met waarden. U kunt bijvoorbeeld geen veld genaamd Prijs maken waarin u meer dan één prijs plaatst. Als u elk snijpunt van een rij en een kolom als een cel beschouwt, mag elke cel slechts één waarde bevatten.

## Tweede normaalvorm

De tweede normaalvorm vereist dat elke niet-sleutelkolom volledig afhankelijk is van de gehele primaire sleutel, en niet slechts van een deel van de sleutel. Deze regel is van toepassing wanneer een primaire sleutel uit meer dan één kolom bestaat. Stel, u hebt een tabel met de volgende kolommen waarin Ordernummer en Product-id de primaire sleutel vormen:

* Ordernummer (primaire sleutel)
* Product-id (primaire sleutel)
* Productnaam

Dit ontwerp schendt de tweede normaalvorm omdat Productnaam afhankelijk is van Product-id, maar niet van Ordernummer, zodat Productnaam niet afhankelijk is van de volledige primaire sleutel. U moet Productnaam uit de tabel verwijderen. Deze kolom hoort namelijk in een andere tabel (Producten).

## Derde normaalvorm

De derde normaalvorm vereist dat niet alleen elke niet-sleutelkolom afhankelijk is van de gehele primaire sleutel, maar ook dat niet-sleutelkolommen onafhankelijk zijn van elkaar.

Een andere manier om dit te formuleren luidt dat elke niet-sleutelkolom afhankelijk moet zijn van de primaire sleutel en alleen van de primaire sleutel. Stel, u hebt een tabel met de volgende kolommen:

* Product-id (primaire sleutel)
* Naam
* Adviesprijs
* Korting

Stel dat Korting afhangt van Adviesprijs. In dat geval schendt deze tabel de derde normaalvorm omdat een niet-sleutelkolom, Korting, afhangt van een andere niet-sleutelkolom, Adviesprijs. Onafhankelijkheid van kolommen betekent dat u elke niet-sleutelkolom moet kunnen wijzigen zonder dat dit van invloed is op andere kolommen. Als u een waarde in het veld Adviesprijs wijzigt, zou Korting echter eveneens veranderen, waardoor deze regel wordt geschonden. In dit geval moet u Korting verplaatsen naar een andere tabel die is gekoppeld aan Adviesprijs.

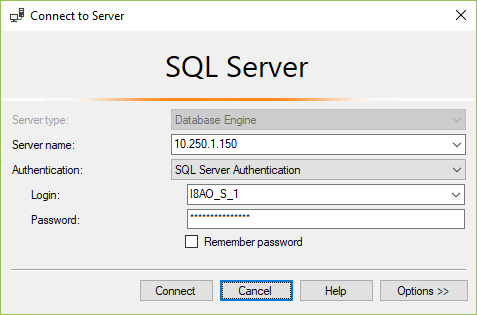
****

**Opdracht**

In het document wordt een database met 5 tabellen als voorbeeld gebruikt (zie blz.12):

* Klanten
* Leveranciers
* Producten
* Orders
* Ordergegevens

Maak met behulp van SQL Server Management Studio deze tabellen aan in jullie database op server 10.250.1.150



Als loginnaam gebruik je **I8AO\_S\_x**, waarbij op de plaats van *x* jullie groepsnummer ingevoerd dient te worden (onder de map SCRUM in OneNote is een overzicht te vinden met de groepsindelingen). Het standaardwachtwoord is **Scrum@ROCvT2018**. Bij de eerste keer inloggen moet je een nieuw wachtwoord instellen. **Zorg ervoor dat dit wachtwoord bij alle groepsleden bekend is.**

De tabellen moet je aanmaken met SQL DDL commando’s. Op de volgende sites kun je leren hoe je het **CREATE TABLE** commando dient te gebruiken:

<https://www.sqltutorial.nl/DDL/sql_create_table>

<https://www.w3schools.com/sql/sql_create_table.asp>

* Je moet zelf geschikte datatypes kiezen voor de velden van de tabellen. Op de volgende site vind je een overzicht van de belangrijkste datatypes:   
  <https://www.w3schools.com/sql/sql_datatypes.asp>  
  (LET OP: ieder DBMS heeft zijn eigen datatypes)
* Maak een zorgvuldige afweging of velden al dan niet verplicht zijn (NOT NULL of NULL)
* Zorg dat iedere tabel een primary key heeft.
* Zorg dat de benodigde relaties tussen de tabellen zijn gedefinieerd.