Leo Wiegerink, Jeanot Bijpost en Marco de Groot

RELATIONELE DATABASES EN SQL

Boom

4e DRUK

Relationele databases en SQL

Vierde, herziene druk

Leo Wiegerink, Jeanot Bijpost, Marco de Groot

met medewerking van
Nikè van Vugt, Harold Pootjes, Bart Pauw

Boom

Met behulp van onderstaande unieke activeringscode krijg je toegang tot de website www.relationeledatabasesensql.nl voor extra materiaal. Deze code is persoonsgebonden en gekoppeld aan de 4e druk. Na activering van de code is de website twee jaar toegankelijk. De code kan tot zes maanden na het verschijnen van een volgende druk geactiveerd worden. De code is eenmalig te gebruiken.

Opmaak binnenwerk: Leo Wiegerink, Amsterdam
Basisontwerp omslag: Dog & Pony, Amsterdam
Omslagontwerp: Coco bookmedia, Amersfoort
Beeld omslag: Pooretat moonsana/Shutterstock

ISBN 9789024429936
ISBN E-book 9789024429943
NUR 173

www.relationeledatabasesensql.nl
www.boomhogeronderwijs.nl

Voorwoord

In deze nieuwe druk worden de relationele theorie en de relationele gegevenstaal SQL als vanouds op een toegankelijke maar gedegen manier behandeld.

Uitgangspunten
Onze aanpak is niet formeel, maar wel conceptueel. Vaak worden daarbij dingen die op het eerste gezicht heel verschillend zijn onder de noemer gebracht van eenzelfde 'rode draad'. De uitleg wordt hierdoor vereenvoudigd en het inzicht verdiept.
'Databasenavigatie' is zo'n rode draad die naar voren komt bij de uitleg van sleutels, joins en subselects, maar ook bij de richtlijnen voor probleemaanpak bij het opstellen van goede SQL-code. Een tweede rode draad is 'normaliseren en denormaliseren'. Zo wordt niet alleen de join maar ook het groeperen in SQL gepresenteerd als een vorm van denormaliseren, wat conceptueel heel zuiver is. Een derde rode draad is 'single point of definition', het principe om wát dan ook op niet meer dan één plek te definiëren. Dat geldt voor databasegegevens zoals plaatsnamen, maar de reikwijdte van het principe blijkt enorm veel uitgebreider.

Bij deze aanpak spreekt het vanzelf dat vraagstellingen conceptueel worden benaderd en niet vanuit de beperkingen van een SQL-dialect. Goed begrip leidt vanzelf tot een kritische benadering, bijvoorbeeld van SQL als relationele taal (dat het maar ten dele is).

Het leren van syntaxis is niet onbelangrijk, maar het ontwikkelen van probleemoplossend vermogen vinden we nog belangrijker, onder het motto 'Het opstellen van SQL-query's is een sport die je kunt leren'. Het stapsgewijs oplossen van vraagstukken, met tussenformuleringen in natuurlijke taal, is een daarbij regelmatig gehanteerde methode.

De theorie wordt ondersteund door vele voorbeelden en opgaven, een krachtig database-managementsysteem, een mooie SQL-omgeving en een bijzondere elektronische leeromgeving.

Doelgroepen
Het boek is geschikt voor informatica- en informatiekundeopleidingen in het hoger onderwijs. Met alle stof op de website erbij is de inhoud voldoende voor een twee- of driesemestercursus. Mede door de elektronische leeromgeving (Boekverkenner) leent het zich ook goed voor zelfstudie. Daarnaast is de Boekverkenner een mooi instrument voor docenten, bij presentaties en instructies.

Nieuw in de vierde druk
In deze vierde druk zijn we overgegaan op de laatste versie van het open source databasemanagementsysteem Firebird, waardoor we nu ook het logisch datatype boolean hebben kunnen gebruiken. Zie 'De software' voor meer informatie.
Op de website is nog allerlei aanvullend materiaal te vinden, waaronder de scripts van de voorbeelddatabases. Er is een speciaal docentengedeelte met onder andere een Powerpoint.

Over de auteurs

Leo Wiegerink studeerde wiskunde aan de Universiteit van Leiden. Als opleider van wiskundeleraren, als informaticadocent aan de Hogere Informaticaopleiding van de Hogeschool van Amsterdam, als docent/cursusontwikkelaar in het bedrijfsleven en ten slotte als cursusontwikkelaar bij de Informatica-faculteit van de Open Universiteit heeft hij ruime ervaring opgedaan met lesgeven aan en cursusontwikkeling voor de doelgroepen van dit boek.

Jeanot Bijpost en Marco de Groot begonnen hun informaticaloopbaan aan de Amsterdamse HIO. Vanuit hun bedrijf, Mattic Software, hebben zij zich beziggehouden met de ontwikkeling en het beheer van grote informatiesystemen. Het ontwikkelen van tools voor systeemontwikkelaars (metasysteemontwikkeling) loopt als rode draad door hun werk.

Dank aan ...

We bedanken iedereen die aan deze of vorige drukken heeft bijgedragen: studenten en docenten voor hun kritische commentaar en niet te vergeten onze achterbannen. Harold Pootjes, Nikè van Vugt en Bart Pauw van de Open Universiteit Nederland voor hun bijdrage aan onder andere de hoofdstukken Security, Transacties en concurrency, Query-optimalisatie, Triggers en stored procedures en De data dictionary, evenals voor de terugkoppeling vanuit het onderwijs. Bijzondere dank gaat uit naar Harold, Bart en Marco: de vele wijzigingen in deze vierde druk zijn grotendeels hun werk.

Leo Wiegerink, Jeanot Bijpost en Marco de Groot
Amsterdam, zomer 2000/voorjaar 2021

De software

De software omvat het open source databasemanagementsysteem Firebird, een interactieve SQL-omgeving IQU (Interactive Query Utility) en een elektronische leeromgeving (Boekverkenner) met een interactieve versie van het volledige boek.

Firebird
Firebird is een krachtige, open source SQL-databaseserver, die zich snel en simpel laat installeren en eenvoudig is in gebruik. Firebird-SQL is bijna volledig conform de SQL-standaard SQL:2008. Firebird omvat een derde-generatietaal voor het programmeren van triggers en stored procedures. We maken gebruik van de laatste versie Firebird 3.0.

Interactive Query Utility
De Interactive Query Utility (IQU®) is een SQL-omgeving, dat wil zeggen een programma om SQL-opdrachten te versturen naar het databasemanagementsysteem. IQU is een zeer handige tool, die vanuit de praktijk is ontwikkeld.

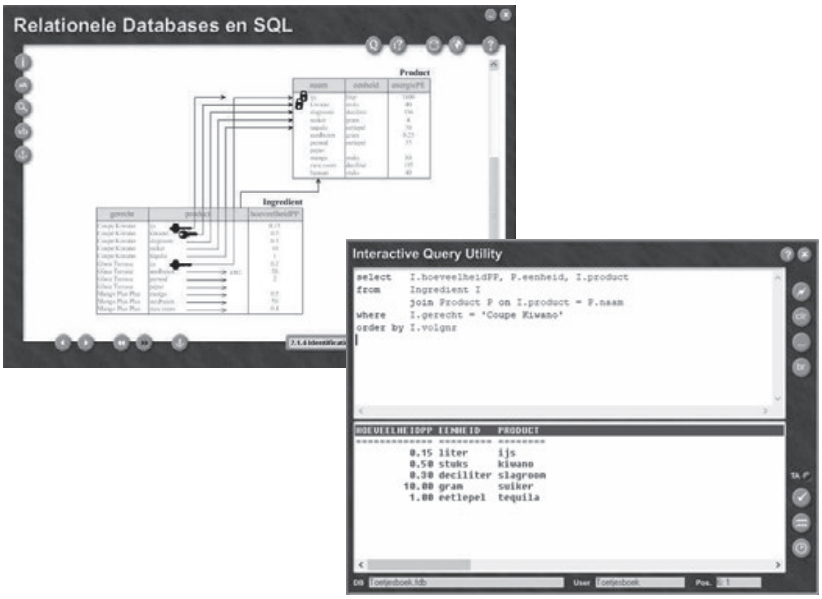
Boekverkenner
De Boekverkenner is een elektronische leeromgeving waarmee het boek op allerlei manieren kan worden benaderd en worden geïntegreerd met het databasemanagementsysteem (Firebird). De Boekverkenner biedt onder meer:

- de volledige tekst van het boek, te benaderen via de inhoudsopgave, een index, een historielijst, bookmarks en een zoekmachine
- de uitwerkingen van alle opgaven
- alle voorbeelddatabases, met hun beschrijving, diagrammen en SQL-scripts
- een eindgebruikersapplicatie met 'zichtbaar' SQL-verkeer
- schakelmogelijkheden tussen tekst, voorbeelddatabases en IQU
- SQL-opdrachten, direct uitvoerbaar vanuit de tekst van het boek
- meting van de databasebelasting van SQL-opdrachten
- verschillende transactiemodellen
- het boek en de Boekverkenner zelf als een van de voorbeelddatabases

Door de snelle schakelmogelijkheden tussen tekst, voorbeelden en IQU, en het moeiteloos installeren en herinstalleren van de voorbeelddatabases, is de Boekverkenner een bijzonder handig en plezierig instrument.

Website
Het boek heeft een website: www.relationeledatabasesensql.nl, met aanvullende informatie en diensten. Er is een algemeen gedeelte met extra materialen, zoals de uitwerkingen van alle opgaven (ook in de Boekverkenner te vinden) en de scripts van de voorbeelddatabases (ook voor Oracle).

Voor het downloaden van de software kunt u hier ook terecht. U hebt hierbij een unieke code nodig, die u vindt op de colofonpagina van dit boek.



Opbouw van het boek

Het boek is opgebouwd uit vier delen en een aantal bijlagen:

Deel A 'Relationele Systemen'
Dit deel behandelt met tal van voorbeelden de theorie van relationele systemen. De hoofdstukken 1 en 2 gaan over de 'achterkant' van die systemen: relationele databases en hun beheerprogramma's. Hoofdstuk 3 geeft een inleiding in de taal (SQL) waarmee met het beheerprogramma wordt gecommuniceerd. Met alle aandacht voor de 'voorkant' van het systeem: de programma's (clients) waarmee eindgebruikers, databasebeheerders of studenten die SQL willen leren het beheerprogramma benaderen. Hoofdstuk 4 gaat over de problematiek van de 'niet-ingevulde waarden' (nulls). Dit deel eindigt met hoofdstuk 5 over de normalisatietheorie, een klassiek database-onderwerp over het tranformeren van ongewenste databasestructuren.

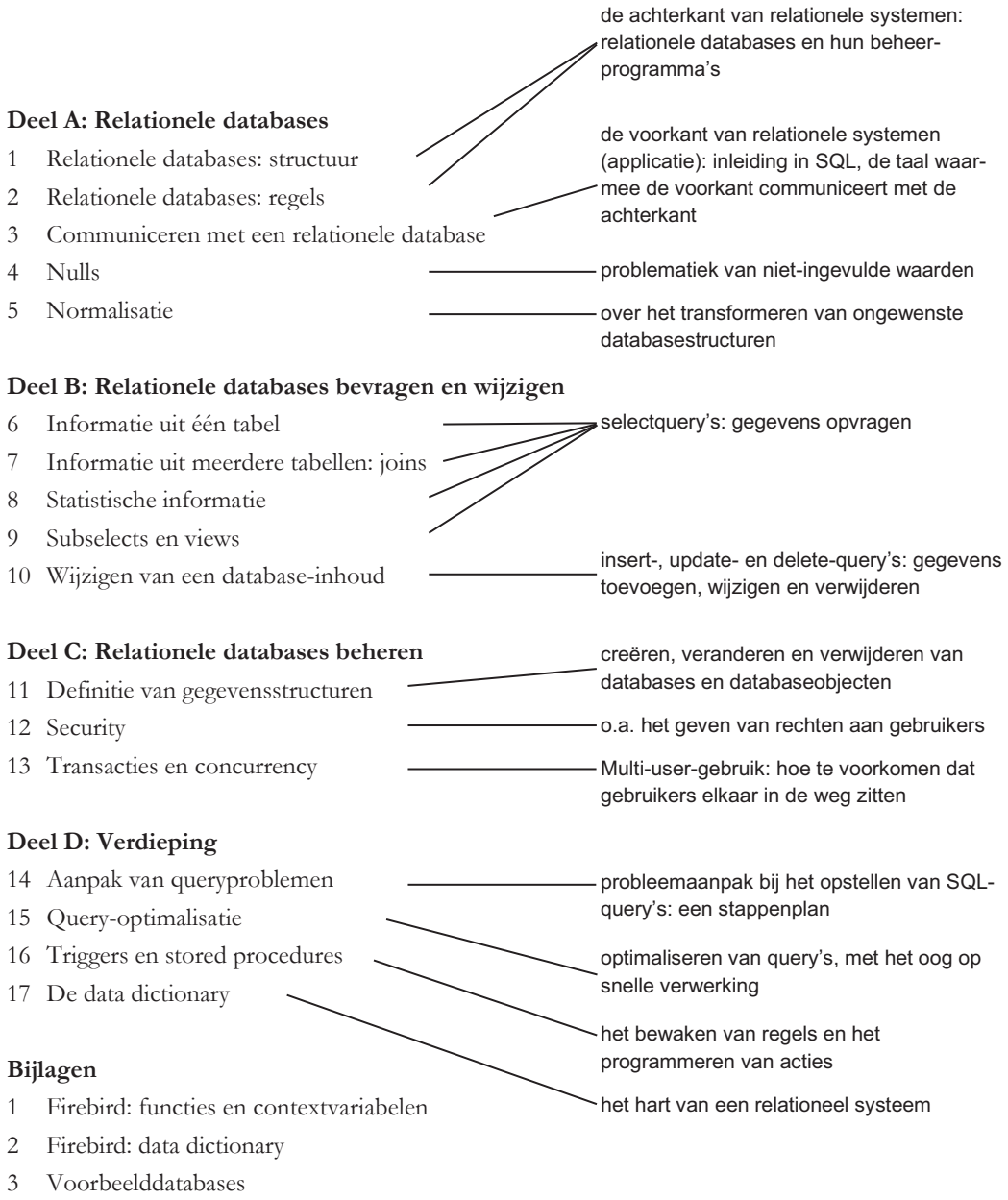
Deel B 'Opvragen en manipuleren van gegevens'
Dit is het eerste deel van een grondige leergang SQL. Het is gewijd aan de SQL-querytaal, de subtaal om gegevens op te vragen en de gegevensinhoud van een database up-to-date te houden. De hoofdstukken 6 tot en met 9 gaan over het opvragen van gegevens, om te voorzien in de informatiebehoeften van gebruikers. De relationele theorie van deel A wordt daarbij optimaal gebruikt om inzicht en verdieping te bereiken. Als vanzelf heeft dit effect op een goede probleemaanpak en een professionele programmeerstijl. Het leidt ook tot een kritische houding ten opzichte van de taal SQL, die om historische en commerciële redenen niet de mooist denkbare relationele taal is. Hoofdstuk 10 behandelt hierna de SQL-commando's om een database-inhoud bij te werken.

Deel C 'Structuur en beheer'
Dit deel vervolgt de SQL-leergang. Hoofdstuk 11 behandelt het creëren, veranderen en verwijderen van databases met hun verschillende typen databaseobjecten. Hierna volgen twee beheerhoofdstukken: hoofdstuk 12 over security, het beschermen van de database en hoofdstuk 13 over multi-user-gebruik, waarin wordt uitgelegd hoe we vermijden dat gelijktijdige gebruikers elkaar in de weg zitten.

Deel D 'Verdieping'
Dit deel bevat vier verdiepende hoofdstukken die stuk voor stuk interessant zijn. Hoofdstuk 14 gaat over probleemaanpak bij het opstellen van SQL-query's. Hoofdstuk 15 behandelt het optimaliseren van query's, met het oog op snelle verwerking. Hoofdstuk 16 gaat over het bewaken van regels en het programmeren van automatische acties via 'triggers en stored procedures' (programmaatjes in een derde-generatie uitbreidingstaal van SQL). Hoofdstuk 17 geeft een inkijkje in het hart van een relationeel systeem: de data dictionary.

Bijlagen
Het gedrukte boek bevat drie bijlagen, met respectievelijk Firebird-specifieke informatie, de Firebird data dictionary en een overzicht van alle voorbeelddatabases.

Inhoudsoverzicht



Inhoud

Deel A:

Relationele databases

1 Relationele databases: structuur

Een relationele database is één component van een groter systeem, een informatiesysteem. De database bevat, gestructureerd in de vorm van tabellen, de gegevens die voor gebruikers van het informatiesysteem de informatie opleveren die zij nodig hebben. Dit hoofdstuk gaat over de structuur van een relationele database en legt daarmee de basis voor al wat volgt.

1.1 Informatiesystemen

Onderwijs-informatiesystemen, een studiefinancierings-informatiesysteem, een belastingdienst-informatiesysteem, tandarts- en huisarts-informatiesystemen, ziekenhuis-informatiesystemen, een antiekdiefstal-informatiesysteem, enzovoort. Er zijn zoveel informatiesystemen dat we allemaal wel ongeveer weten wat het zijn: 'systemen waar je informatie in kunt stoppen en informatie uit kunt halen'. We zullen de twee aspecten: *informatie* en *systeem* eens wat preciezer beschouwen.

Informatie heeft te maken met *gegevens* die ergens zijn opgeslagen: op papier, in een computergeheugen of op een andere gegevensdrager. Die gegevens kunnen bestaan uit tekst of getallen, uit grafieken of andere plaatjes. Ook gesproken tekst op een bandje kun je opvatten als een verzameling gegevens. Informatie is echter méér dan die gegevens: het heeft te maken met de *betekenis* die aan de gegevens moet worden gehecht. En gegevens kunnen alleen maar wat betekenen als ze iets uitdrukken over een 'wereld'. Bijvoorbeeld een wereld van patiënten, doctoren, medische ingrepen enzovoort in een ziekenhuis. Of een wereld van antiek, diefstal en opsporing. Of een heel klein wereldje van recepten voor lekkere toetjes: het geautomatiseerde Toetjesboek dat we verderop als voorbeeld zullen gebruiken. Informatie drukt *feiten* uit over zo'n wereld.

Een *systeem* is iets dat geordend is, waar een bepaalde structuur in zit. Een *informatiesysteem* is dus een systeem waarin informatie (betekenisvolle gegevens over een bepaalde 'wereld') gestructureerd is opgeslagen. Daarnaast is het een instrument voor gebruikers van het systeem, die er informatie uit moeten kunnen opvragen om antwoorden te krijgen op relevante vragen over de 'wereld' in kwestie.

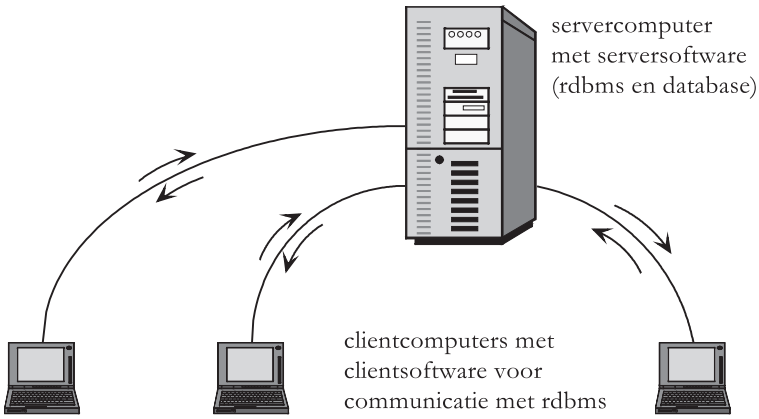
Een informatiesysteem hoeft niet geautomatiseerd te zijn, er hoeft helemaal geen computer aan te pas komen. Een verzameling kaartenbakken kan een betrouwbaar informatiesysteem vormen, als het goed wordt beheerd. Maar zo'n papieren systeem heeft beperkingen: het opzoeken kan lang duren en koppelen met andere systemen is veelal onuitvoerbaar. Een geautomatiseerd systeem kent dit soort beperkingen in mindere mate. Maar alleen als het goed is ontworpen.

1.2 Relationele informatiesystemen

Veel moderne informatiesystemen maken gebruik van een manier van gegevensopslag die we *relationeel* noemen. Dit houdt – informeel uitgedrukt – in dat de gegevens in de vorm van *tabellen* zijn opgeslagen. Alle tabellen bij elkaar, doorgaans behorend bij één 'wereld', vormen een *relationele database*. Een *relatie* is een wiskundige structuur die veel overeenkomst vertoont met wat we in het dagelijks leven een tabel noemen, vandaar de term 'relationeel'. En *database* betekent 'gegevensbank', ofwel opslagplaats voor gegevens.

SQL-server en SQL-clients

Bij grotere informatiesystemen met meerdere gebruikers is er meestal één centrale computer (de *server*) waarop de database wordt bewaard en beheerd. Elke gebruiker van het systeem beschikt dan over een pc of vergelijkbare computer (de *client*) waarop een *applicatie* (toepassingsprogramma) draait. Deze configuratie, waarbij een taakverdeling bestaat tussen een beheerprogramma op de server-computer en programma's op de clients, is de meest voorkomende fysieke verschijningsvorm van wat *client/server-architectuur* wordt genoemd, zie figuur 1.1.
Het serverprogramma heet *relationeel databasemanagementsysteem* (*rdbms*), ofwel 'relationeel gegevensbankbeheersysteem'.

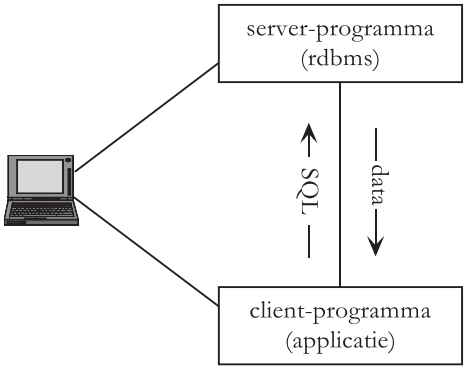


Figuur 1.1 Client/server-architectuur met pc's en centrale databaseserver

De server is doorgaans met de clients verbonden door een netwerkverbinding. De gebruikers sturen via hun applicaties opdrachten naar het rdbms. Die opdrachten worden geformuleerd in een *gegevenstaal*. Voor relationele systemen is de meest gebruikte gegevenstaal SQL. Het rdbms wordt daarom ook vaak een SQL-*server* genoemd, en de applicatie een SQL-*client*. Deze voert de opdrachten uit, bijvoorbeeld het wijzigen van een adres of het terugsturen van een overzicht met gegevens. Een eindgebruiker hoeft de taal SQL niet te kennen. Hun client 'spreekt' SQL zonder dat ze het merken. Maar ontwikkelaars en beheerders van relationele databases dienen een gedegen kennis van SQL te hebben. Daarom is het grootste deel van dit boek daaraan gewijd.

Bij kleine systemen kunnen het rdbms en een applicatie op één computer draaien, bijvoorbeeld op een pc of een notebookcomputer. Al is de taakverdeling tussen serverprogramma (rdbms) en clientprogramma (applicatie) dan minder zichtbaar, in feite is er niets wezenlijks veranderd, zie figuur 1.2. Wanneer je de voorbeelden en opgaven in dit boek zelf uitprobeert, zal dat waarschijnlijk op één machine gebeuren, ofwel: met client en server in één kastje.

SQL-dialecten
Hoewel standaardisatiecommissies de ontwikkeling van SQL hebben proberen te sturen (in hoofdstuk 3 vertellen we daar meer over) hebben de rdbms'en van verschillende leveranciers elk hun eigen SQL-dialect. Bij dit boek is Firebird meegeleverd, een open sourceproduct. Dit volgt zeer goed de belangrijke SQL:2008-standaard. Hiermee kunt u de voorbeelden uitproberen en de opdrachten maken. Via de elektronische leeromgeving (Boekverkenner) wordt het u daarbij heel erg gemakkelijk gemaakt.



Figuur 1.2 Client/server op één computer

Een stukje geschiedenis
In 1970 verscheen in de *Communications of the Association for Computing Machinery* een artikel van IBM-onderzoeker dr. E.F. Codd (1970): 'A Relational Model of Data for Large Shared Data Banks'. Deze dr. Codd wordt sedertdien steevast aangeduid als de 'vader van het relationele model'. In dit artikel wordt een theorie uiteengezet, gebaseerd op de wiskundige verzamelingenleer, over het opslaan en manipuleren van gegevens door middel van tabelstructuren.
Het artikel bracht veel onderzoek op gang, met als doel een 'relationeel' alternatief te ontwikkelen voor de toen bestaande databasearchitecturen, waar men om verschillende redenen niet tevreden over was. IBM zelf startte een research project 'System/R', dat beoogde een werkend relationeel systeem op te leveren. Halverwege de 70'er jaren resulteerde dit in een prototype van een rdbms, dat in de jaren daarna verder werd ontwikkeld.
Onderzoek naar relationele vraagtalen gaf het leven aan onder meer de taal SEQUEL, wat stond voor Structured English Query Language ('gestructureerde Engelse vraagtaal'). Later werd deze naam om auteursrechtelijke redenen gewijzigd in SQL, Structured Query Language, wat vaak nog steeds als 'seOnline' wordt uitgesproken.

Niet IBM zelf, maar een bedrijf genaamd 'Relational Software, Inc.' bouwde het eerste commerciële rdbms gebaseerd op SQL. Dit product, Oracle, kwam uit in 1979. Het bedrijf heet inmiddels al heel lang Oracle Corporation en is marktleider voor rdbms'en voor client/server-systemen.

IBM zelf heeft zich een dominante positie verworven op de mainframemarkt. Het meest bekende IBM-rdbms is Database 2 (DB/2) geworden.

Vermeldenswaardig is ook het bedrijf 'Relational Technology, Inc.' dat in 1981 een commercieel SQL-rdbms het leven liet zien, genaamd Ingres. Het prototype van Ingres en de eerste commerciële versies van Ingres waren niet gebaseerd op SQL maar op de taal QUEL, die veel minder dan SQL op Engels lijkt en een strakkere, logische structuur heeft. Wie in dit boek wat verder gevorderd is, zal het misschien met ons betreuren dat QUEL uiteindelijk het onderspit heeft moeten delven tegenover SQL.
In de jaren negentig heeft ook Microsoft zich op de databasemarkt begeven. Momenteel is Microsoft met Ms Sql Server een van de belangrijke leveranciers. Een ander bekend Microsoft-product is Ms Access dat echter maar beperkte mogelijkheden heeft en minder geschikt is voor grote databases in complexe omgevingen.

Client/server-architectuur

Een zuivere benadering van de *client/server-architectuur* gaat niet uit van een fysieke hardwareconfiguratie, maar van een rolverdeling tussen programma's. Daarbij verleent het ene programma (het serverprogramma) diensten aan een ander programma dat om die diensten vraagt (het clientprogramma). In de praktijk wordt de term vooral gebruikt voor een netwerkconfiguratie met aan de 'achterkant' een databaseserver en aan de 'voorkant' client-pc's met grafisch georiënteerde applicatieprogrammatuur. Belangrijk hierbij zijn de softwarecomponenten die garanderen dat de gegevens aan bepaalde *bedrijfsregels* voldoen, of die automatisch bepaalde acties veroorzaken. Tezamen worden deze vaak aangeduid met de wat verwarrende termen *logica* of *business logic*.

Fat clients of thin clients
De bedrijfslogica staat in principe volledig of bijna volledig op de server. Zoals we later zullen zien: als onderdeel van de database. Daarnaast kunnen ook al de nodige controles op de clients plaatsvinden. Denk bijvoorbeeld aan controles op geldige postcodes of e-mailadressen. Dat heeft als voordeel dat het netwerkverkeer wordt beperkt, immers de gegevens die worden verstuurd zijn dan al *gevalideerd* (getoetst aan de regels en in orde bevonden). Een nadeel kan zijn dat de applicaties sneller bijgewerkt moeten worden (versiebeheer op de clients).
Bij een *fat client* worden veel regels al in het clientprogramma bewaakt. Bij een *thin client* is dat niet of nauwelijks het geval. Daarbij is in het algemeen meer netwerkverkeer nodig, immers alle invoerpogingen worden pas op de server gecontroleerd. Doordat echter de bedrijfslogica voornamelijk op de server wordt beheerd, wordt het applicatiebeheer eenvoudiger.
In een enkel geval vindt op *alle* applicaties uitgebreide regelbewaking plaats. Het is dan niet nodig om dit ook nog eens op de server te doen. We hebben dan een *thin server* (tegenover, in het normale geval, een *fat server*). Een belangrijk bijkomend voordeel van zo'n thin server is dat er slechts de meest elementaire SQL-commando's nodig zijn om de database aan te maken en te onderhouden. Die elementaire commando's zijn voor vrijwel alle SQL-dialecten gelijk, waardoor de applicaties met slechts geringe aanpassingen geschikt kunnen worden gemaakt voor rdbms'en van andere leveranciers (Oracle, Microsoft, IBM, ...).

Wanneer men het over client/server heeft, wordt meestal verondersteld dat de applicatieprogramma's tenminste een deel van de business logic voor hun rekening nemen. Dat was immers juist de winst ten opzichte van de voorgangertechnologie met mainframes.

Van mainframe naar client/server
De client/server-architectuur is de opvolger van een architectuur met één krachtige, centrale computer, mainframe genaamd, met 'domme' terminals. De terminals dienden uitsluitend voor weergave van resultaten en invoer van gegevens. Alle verwerking, zelfs de kleinste controle op correctheid van gegevens, vond op het mainframe plaats. Je zou kunnen zeggen: een extreem geval van een thin client. De stap naar client/server was een grote vooruitgang, vanwege de beperking van het netwerkverkeer. We moeten hierbij bedenken dat in de tijd waarin dit speelt (de jaren '70) de *bandbreedte* van netwerken (de capaciteit in bits per tijdeenheid) gering was.

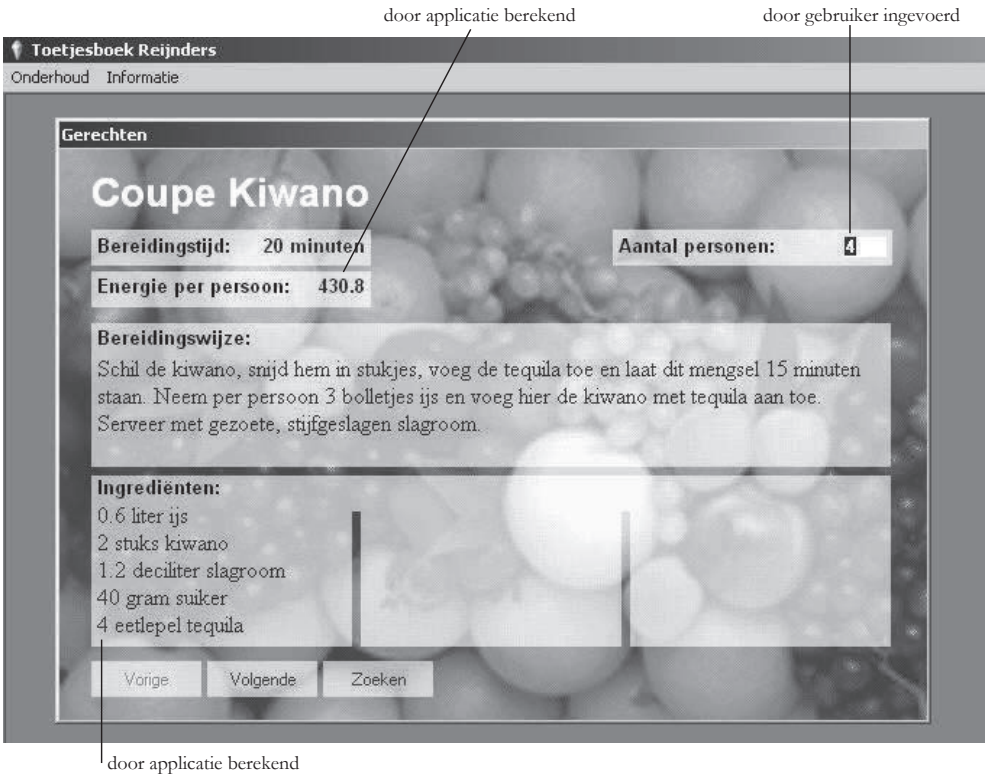
Van client/server naar internet
De opkomst van internet heeft nog niet het einde ingeluid van client/server, maar heeft wel de thin client in ere hersteld. Het ligt wat subtiel: aan de clientkant is het eigenlijke applicatieprogramma nu de internetbrowser, die niet bepaald 'thin' is. Maar met het oog op een specifieke toepassing kunnen de via een webserver verstuurde webpagina's als applicatie worden beschouwd. Denk bijvoorbeeld aan een pagina met een bestelformulier. Hoewel dergelijke formulieren steeds intelligenter worden (door meer en meer ingebouwde logica, vaak met Javascript-programmacomponenten) zijn ze nog steeds als thin clients te beschouwen. Maar de nadelen van de vroegere thin clients gelden hier veel minder: de bandbreedte van het netwerkverkeer is enorm toegenoemen en vooral: de applicaties zijn zelf gegevens geworden die worden overgestuurd. Het beheerprobleem, voor zover samenhangend met verspreide locaties, bestaat niet meer.

Minstens zo belangrijk als deze zuiver technische kwesties is de toegankelijkheid van serverapplicaties: nu ook van buiten een bedrijfsnetwerk. Behalve voor de eigen bedrijfsmedewerkers kunnen internetapplicaties ook voor externe doelgroepen (klanten of leveranciers!) worden ontwikkeld. De communicatie met een (relationele) database vindt in al deze gevallen plaats via een aantal softwarelagen (*tiers*, uit te spreken met een 'ie'), met aan de ene kant de clientlaag (internetbrowser) en aan de andere kant het rdbms. En het rdbms wordt nog steeds 'toegesproken' in SQL, maar de SQL-opdrachten worden nu verstuurd via een tussenlaag aan de serverkant.

1.3 Reijnders' Toetjesboek

In dit hoofdstuk en in beide volgende hoofdstukken staat één voorbeeldinformatiesysteem centraal: Reijnders' Toetjesboek. Dit is een interactief receptenboek voor lekkere toetjes, op een computer die bij voorkeur is ingebouwd in het keukenaanrecht. Gezien deze 'huiselijke' context is het aannemelijk dat op die computer een lokale versie van het relationele databasemanagementsysteem (rdbms) draait. In principe echter kan het rdbms ook elders draaien, op een externe server. De applicatie op de 'keukenclient' communiceert daar dan mee via een netwerkverbinding.
In Reijnders' Toetjesboek kunt u recepten van toetjes opzoeken, u kunt ze veranderen en u kunt er uw eigen recepten aan toevoegen. U kunt zoeken volgens allerlei criteria, bijvoorbeeld snel klaar te maken en/of caloriearm en/of met aardbeien. Uit de ingrediëntinformatie wordt automatisch afgeleid hoeveel energie per persoon elk gerecht bevat, uitgedrukt in kilocalorieën (kcal).
Er zijn beperkingen. Zo worden de ingrediënten in Reijnders' Toetjesboek altijd gemeten in dezelfde maateenheid, ongeacht het gerecht. Bijvoorbeeld melk altijd in liters en dus nooit in eetlepels. Verder is de opgegeven bereidingstijd een vaste waarde, ongeacht het aantal personen.

Figuur 1.3 toont een receptvenster, met informatie over het gerecht Coupe Kiwano. Dit is een voorbeeld van hoe een clientprogramma (applicatie) informatie op een grafische manier aan een gebruiker kan presenteren. De meeste van de gepresenteerde gegevens zijn opgehaald uit de (relationele) database, waar ze in verschillende tabellen zijn opgeslagen. Een aantal echter zijn *berekende waarden*, die niet zijn opgeslagen, maar door de applicatie worden berekend. Eén waarde is ingevuld door de gebruiker.



Figuur 1.3 Receptvenster voor het gerecht Coupe Kiwano

Toelichting
- Het aantal personen kan door de gebruiker worden ingesteld; de initiële waarde is 4.
- De ingrediënthoeveelheden van een gerecht worden door de applicatie berekend uit de opgeslagen (maar niet afgebeelde) ingrediënthoeveelheden per persoon en het aantal personen.
- De energie per persoon van een gerecht wordt door de applicatie berekend uit de opgeslagen ingrediënthoeveelheden per persoon en de eveneens opgeslagen energiewaarden per eenheid.

In de volgende paragrafen zullen we stap voor stap achterhalen wat de structuur is van de databasetabellen. Gaandeweg zullen we dan vertrouwd raken met de belangrijkste begrippen uit het *relationele model:* de theorie over tabellen, de wijze waarop deze onderling samenhangen en de regels waaraan zij moeten voldoen.

Opgave 1.1
De energie per persoon van een gerecht is een afgeleide waarde die door het rdbms zelf wordt berekend uit de energiewaarden van de ingrediënten (per maateenheid) en de hoeveelheden per persoon. Reken dit (uitgaande van figuur 1.3) na voor Coupe Kiwano, als gegeven is dat in de database de volgende (in het venster niet zichtbare) energiewaarden zijn opgeslagen:

ijs	1600 kcal per liter
kiwano	40 kcal per stuk
slagroom	336 kcal per deciliter
suiker	4 kcal per gram
tequila	30 kcal per eetlepel

Opmerking: de uitwerking van deze opgave en van alle andere opgaven vindt u in de Boekverkenner.

1.4 Alle gegevens in één tabel

In deze paragraaf zullen we een opslagstructuur bespreken (en verwerpen) waarbij alle gegevens in één tabel worden ondergebracht. Daarbij komen enkele belangrijke begrippen aan de orde: *redundantie* en *herhalende groep*.

Eerste stap naar relationele opslagstructuur

Figuur 1.4 bevat drie tabellen met gerechtinformatie: één voor Coupe Kiwano, één voor Glace Terrace en één voor Mango Plus Plus.

Per recept zijn vijf kenmerken opgenomen:
– de naam van het gerecht
– de energie per persoon (in kcal)
– de bereidingstijd
– de bereidingswijze
– de ingrediëntinformatie

De eerste vier hiervan zijn *enkelvoudige gegevens*: tekst of getalwaarden. In andere gegevensverzamelingen zijn ook andere enkelvoudige waarden mogelijk, zoals kalenderdata, of een foto of videofragment. Het criterium voor 'enkelvoudig gegeven' is dat het rdbms zich niet hoeft bezig te houden met de interne structuur ervan.

gerecht	Coupe Kiwano			
energie per persoon (kcal)	431			
bereidingstijd (minuten)	20			
bereidingswijze	Schil de kiwano, snijd hem in stukjes, voeg de tequila toe en laat dit mengsel 15 minuten staan. Neem per persoon 3 bolletjes ijs en voeg hier de kiwano met tequila aan toe. Serveer met gezoete, stijfgeslagen slagroom.			
ingrediënten	product	hoeveelheid per persoon	eenheid	energie per eenheid (kcal)
	ijs	0.15	liter	1600
	kiwano	0.5	stuks	40
	slagroom	0.3	deciliter	336
	suiker	10	gram	4
	tequila	1	eetlepel	30

gerecht	Glace Terrace			
energie per persoon (kcal)	403			
bereidingstijd (minuten)	5			
bereidingswijze	Neem drie bolletjes ijs, voeg hieraan de gesneden aardbeien toe, besprenkel dit rijkelijk met pernod en maak dit bijzondere gerecht af met versgemalen peper.			
ingrediënten	product	hoeveelheid per persoon	eenheid	energie per eenheid (kcal)
	ijs	0.2	liter	1600
	aardbeien	50	gram	0.25
	pernod	2	eetlepel	35
	peper			

gerecht	Mango Plus Plus			
energie per persoon (kcal)	131			
bereidingstijd (minuten)	8			
bereidingswijze	Snijd de - geschilde - mango in stukjes, meng deze met de gehalveerde aardbeien, en serveer dit met de zure room.			
ingrediënten	product	hoeveelheid per persoon	eenheid	energie per eenheid (kcal)
	mango	0.5	stuks	80
	aardbeien	50	gram	0.25
	zure room	0.4	deciliter	195

Figuur 1.4 Schematische weergave van drie receptvensters

Het vijfde kenmerk, de ingrediëntinformatie, is *samengesteld*: het is niet één waarde, maar een hele *subtabel* van waarden. Elke rij van zo'n subtabel bevat gegevens over één ingrediënt in het betreffende gerecht:
- de product van het ingrediënt
- de hoeveelheid per persoon
- de maateenheid waarin het product wordt gemeten
- een kolom 'energie per eenheid', gemeten in kcal

Merk, afgezien van de subtabelweergave, de volgende verschillen op met figuur 1.3.
- Het 'aantal personen' is verdwenen; de waarde hiervan wordt immers niet in de database bewaard.

– De 'hoeveelheden' (voor het boodschappenlijstje) komen niet meer voor. Deze worden door de applicatie berekend en hoeven dus niet te worden opgeslagen. De hoeveelheden per persoon zijn ervoor in de plaats gekomen. Hieruit en uit het aantal personen dat door een gebruiker wordt ingevuld kan de benodigde hoeveelheid van een ingrediënt worden berekend.
– Bij de producten is de 'energie per eenheid' erbij gekomen; de Toetjesboek-applicatie heeft deze nodig om de energie per persoon van elk gerecht te kunnen berekenen.

In tegenstelling tot de hoeveelheden wordt de energie per persoon wel in de database opgeslagen. Het rdbms bevat een mechanisme waardoor ze automatisch worden herberekend als dat nodig is.

De 'peper' in Glace Terrace heeft wat bijzonders: de hoeveelheid ontbreekt, evenals de eenheid en de energie per eenheid. Welnu, dat kan: dat is gewoon 'naar smaak'!

Afleidbare gegevens: in database of applicatie?

Het was mogelijk geweest om de hoeveelheden en de energiewaarden per persoon gelijk te behandelen:
– allebei opslaan in de database (met een mechanisme om hun waarden te laten herberekenen als dat nodig is)
– allebei berekenen in de applicatie (vlak voor ze worden getoond)

Bewust hebben we het verschillend gedaan om beide mogelijkheden te illustreren. De vraag wanneer in de praktijk voor het een of voor het ander wordt gekozen, laten we in het midden.

Eén tabel met subtabellen

Alle genoemde kenmerken voor de drie gerechten kunnen we onderbrengen in één grotere tabel, met drie *rijen* en vijf *kolommen*, zie figuur 1.5. Merk op dat alle gegevens van één gerecht nu in één rij staan, en overeenkomstige kenmerken van gerechten in één kolom. Elke kolom heeft een *kolomnaam*.
Door deze wijze van weergeven hebben we bereikt dat de *logische rijstructuur* overeenkomt met de *fysieke rijstructuur*. Dat wil zeggen dat een logische rij (de gegevens over één 'ding') horizontaal wordt getekend en een logische kolom (overeenkomstige gegevens van de 'dingen') verticaal.

Enkele kolomkoppen zijn wat compacter genoteerd: energiePP voor 'energie per persoon', energiePE voor 'energie per eenheid' en hoeveelheidPP voor 'hoeveelheid per persoon'.

Het is belangrijk om de samengestelde kolom ingrediënten als één enkele kolom te zien!

Het 'snijpunt' van een rij en een kolom heet een *cel*. De cellen in de kolommen gerecht, energiePP, bereidingstijd en bereidingswijze bevatten een enkelvoudige waarde.
De cellen in de kolom ingrediënten bevatten een samengestelde waarde: elke cel bevat een *subtabel* met vier kolommen. De kolom ingrediënten is dan ook de enige kolom waarvan de naam een meervoudsvorm is.
De lichtgrijze achtergrond in kolom energiePP duidt aan dat de gegevens in deze kolom afleidbaar zijn.
Merk op dat de kolom met gerechtnamen 'naam' heet en niet 'gerecht'. In het algemeen geven we een kolom nooit dezelfde naam (ongeacht hoofd- of kleine letter) als een tabel. Met de kolomnaam 'naam' drukken we uit dat een gerecht een naam heeft. (Een gerecht heeft geen gerecht!)

Gerecht

naam	energiePP	bereidingstijd	bereidingswijze	ingrediënten			
				product	hoeveelheidPP	eenheid	energiePE
Coupe Kiwano	431	20	Schil ...	ijs kiwano slagroom suiker tequila	0.15 0.5 0.3 10 1	liter stuks deciliter gram eetlepel	1600 40 336 4 30
Glace Terrace	403	5	Neem ...	ijs aardbeien pernod peper	0.2 50 2	liter gram eetlepel	1600 0.25 35
Mango Plus Plus	131	8	Snijd ...	mango aardbeien zure room	0.5 50 0.4	stuks gram deciliter	80 0.25 195

Figuur 1.5 Alle gegevens in één tabel, met drie rijen en vijf kolommen

Is dit nu een geschikte tabelstructuur om de gegevens in de database op te slaan? Nee, want er kleven maar liefst drie bezwaren aan opslag volgens deze structuur:

Bezwaar 1: Bepaalde informatie wordt meer dan één keer opgeslagen.
Bezwaar 2: Een tabelstructuur met subtabellen maakt het databasebeheer erg ingewikkeld.
Bezwaar 3: Gegevens over producten worden alleen in de context van gerechten opgeslagen, terwijl producten een zelfstandig belang kunnen hebben.

Meervoudige informatieopslag (*redundantie*), kolommen met subtabellen (*herhalende groepen*) en het principe dat alle 'soorten dingen' een gelijkwaardige behandeling verdienen, worden in het vervolg van dit hoofdstuk uitgebreid behandeld.

Redundantie

Op twee plaatsen wordt vermeld dat ijs wordt gemeten in 'liter'. Dit is vaker dan nodig; producten worden immers in Reijnders' Toetjesboek altijd in dezelfde eenheid gemeten. IJs bijvoorbeeld in liter en aardbeien in gram. Wanneer u dus bij Coupe Kiwano leest dat ijs in liter wordt gemeten, weet u zonder verder te kijken dat dit ook voor Glace Terrace geldt, en omgekeerd. We noemen dat *redundantie*, ofwel *redundante gegevensopslag*. Redundant betekent overtollig: bepaalde gegevens zijn afleidbaar uit andere gegevens in de database. Zou u in de rij van Glace Terrace 'liter' bij 'ijs' wegstrepen, dan kon u het reconstrueren door bij Coupe Kiwano de eenheid van 'ijs' op te zoeken. Zie figuur 1.6. Die reconstrueerbaarheid uit andere gegevens is hét kenmerk van redundante opslag. Ook de kolom energiePE geeft aanleiding tot redundantie: van ijs wordt twee keer vermeld wat de energiewaarde is.

Gerecht

naam	energiePP	bereidingstijd	bereidingswijze	ingrediënten			
				product	hoeveelheidPP	eenheid	energiePE
Coupe Kiwano	431	20	Schil ...	ijs kiwano slagroom suiker tequila	0.15 0.5 0.3 10 1	liter stuks deciliter gram eetlepel	1600 40 336 4 30
Glace Terrace	403	5	Neem ...	ijs aardbeien pernod peper	0.2 50 2	gram eetlepel	0.25 35

Figuur 1.6 Redundante gegevensopslag: reconstrueerbaarheid uit de context

Redundante opslag kost extra opslagruimte, maar dat is hier nauwelijks een bezwaar. Een echt bezwaar is het gevaar dat de gegevens onderling niet meer kloppen. Stel bijvoorbeeld dat bij Coupe Kiwano staat dat ijs 1600 kilocalorieën per maateenheid bevat, en bij Glace Terrace dat het 1450 is, zie figuur 1.7. Een van de twee moet dan fout zijn. We spreken van *inconsistentie* ofwel *inconsistente gegevensopslag*. Inconsistente gegevens zijn gegevens waarvan de betekenissen onderling tegenstrijdig zijn. Redundantie brengt vaak het gevaar van inconsistentie met zich mee. In de volgende paragraaf zullen we zien dat dit niet altijd het geval is.

Gerecht

naam	energiePP	bereidingstijd	bereidingswijze	ingrediënten			
				product	hoeveelheidPP	eenheid	energiePE
Coupe Kiwano	431	20	Schil ...	ijs kiwano slagroom suiker tequila	0.15 0.5 0.3 10 1	liter stuks deciliter gram eetlepel	1600 40 336 4 30
Glace Terrace	403	5	Neem ...	ijs aardbeien pernod peper	0.2 50 2	liter gram eetlepel	1450 0.25 35

Figuur 1.7 Inconsistente gegevensopslag

Merk op dat we in het voorgaande enerzijds spreken over *gegevens*, anderzijds over *informatie*. Bij gegevens gaat het om afzonderlijke waarden, bij informatie om de betekenis van die waarden, die veelal alleen is uit te drukken in combinatie met andere waarden. Hoewel redundantie altijd te maken heeft met de betekenis van gegevens, zullen we toch ook wel spreken over *redundante gegevens*. Dat zijn dan gegevens die reconstrueerbaar zijn uit andere gegevens, op grond van betekenissen. In paragraaf 1.6 gaan we dieper in op het onderscheid tussen gegevens en informatie.

Gecontroleerde redundantie

Behalve de kolommen eenheid en energiePE bevat figuur 1.6 nóg een kolom die aanleiding geeft tot redundantie: de kolom energiePP, die de energie per persoon van een gerecht aangeeft, gemeten in kilocalorieën. Deze kolom bevat zelfs uitsluitend redundante gegevens; de waarden worden immers berekend uit energiePE en hoeveelheidPP (hoe precies?). Dit is een geval van *gecontroleerde redundantie*: het is de bedoeling dat de gebruiker de berekende energiewaarde niet kan veranderen. Daardoor bestaat er geen gevaar voor inconsistente gegevens.

Het mechanisme om de juiste waarde steeds opnieuw te berekenen is gebaseerd op *triggers*, een bepaald type programmaatjes waarvan de code in de database is opgenomen. Zou de gebruiker bijvoorbeeld een hoeveelheid per persoon veranderen, dan zal het systeem automatisch de juiste trigger aanroepen en deze een nieuwe waarde voor de energiewaarde van het gerecht laten berekenen. Wijzigt de gebruiker een energiewaarde van een product, dan zal automatisch een andere trigger worden aangeroepen, die de energiewaarden per persoon aanpast van alle gerechten waar dat product in voorkomt. Het programmeren van triggers (in een programmeertaal die een uitbreiding biedt op de mogelijkheden van SQL) wordt besproken in hoofdstuk 16.

Herhalende groepen

Redundantie was één probleem van de tabelstructuur van figuur 1.5. Het tweede probleem was de kolom 'ingrediënten', die voor elke gerechtrij een subtabel met ingrediëntgegevens bevat. Zo'n samengestelde kolom, die per rij van de hoofdtabel een subtabel bevat, wordt een *herhalende groep* genoemd (Engels: *repeating group*). Met 'groep' worden de kolommen van de subtabellen bedoeld. Het predikaat 'herhalend' is helaas wat ongelukkig gekozen.

Herhalende groepen en redundantie

Wat is er eigenlijk mis met een herhalende groep? Op het eerste gezicht lijkt het dat de herhalende groep verantwoordelijk is voor de redundantie in de kolommen eenheid en energiePE. Dit is echter maar schijn, want we zullen nog zien dat we de tabel op twee manieren kunnen omvormen:

- tot twee gerelateerde tabellen zonder redundantie, maar met herhalende groep;
- tot twee gerelateerde tabellen met redundantie, maar zonder herhalende groep.

Bezwaren van herhalende groepen

'Herhalende groep' en 'redundantie' staan dus los van elkaar. Echte bezwaren van herhalende groepen zijn:

- het databasebeheer wordt er veel ingewikkelder door
- ze leiden tot een ingewikkelder databasestaal
- gebrek aan symmetrie: het ene 'soort ding' wordt heel anders behandeld dan het andere 'soort ding'

Ingewikkelder databasebeheer

Zouden de Toetjesboek-gegevens volledig per gerecht worden opgeslagen, met een herhalende groep van ingrediënten, dan zou het invoegen van nieuwe rijen op twee niveaus moeten gebeuren, zie figuur 1.8:

1. binnen een subtabel, bij het invoegen van een nieuw ingrediënt bij een bestaand gerecht
2. binnen de hoofdtabel, bij het invoegen van een nieuw gerecht; hierbij moeten de gewone, enkelvoudige gerechtgegevens worden ingevoerd, maar ook een subtabel van ingrediënten, wat een heel ander soort bewerking is

Gerecht

naam	energiePP	bereidingstijd	bereidingswijze	ingrediënten			
				product	hoeveelheidPP	eenheid	energiePE
Coupe Kiwano	**491**	20	Schil ...	ijs	0.15	liter	1600
				kiwano	0.5	stuks	40
				slagroom	0.3	deciliter	336
				suiker	10	gram	4
				tequila	1	eetlepel	30
				topaz	**1**	**stuks**	**60**
Glace Terrace	...	...	...	...	...	...	...
Mango Plus Plus	...	...	...	...	...	...	...
Macedoine	**175**	**10**	**Snijd ...**	**mango**	**0.5**	**stuks**	**80**
				tangarine	**1**	**stuks**	**60**
				cherimoya	**1**	**stuks**	**75**

toevoegen rij (met subtabel) in hoofdtabel

toevoegen rij in subtabel

Figuur 1.8 Invoegen van rijen op twee niveaus

Ingewikkelder gegevenstaal

Ook het verwijderen van rijen wordt ingewikkelder: het verwijderen van een rij in een subtabel is een ander soort bewerking dan het verwijderen van een volledige gerechtrij. Net zoiets geldt voor het wijzigen van gegevens en het opvragen van overzichten. We zullen nog zien dat al die bewerkingen en opvragingen worden gerealiseerd door opdrachten die zijn geformuleerd in een *gegevenstaal*. Die taal zou een stuk ingewikkelder worden wanneer we tabellen met herhalende groepen toelieten.

Gebrek aan symmetrie

Producten die in gerechten als ingrediënt voorkomen, worden alleen in de context van een gerecht opgeslagen. Dat is jammer, want ze hebben ook een belang van zichzelf. Je zou bijvoorbeeld een calorieëntabel kunnen samenstellen uit alle productinformatie, los van gerechten. Een neutrale, contextvrije benadering van de verschillende 'soorten van dingen' zal een van de belangrijkste kenmerken blijken te zijn van een goedgestructureerde relationele database. Later zullen we hieraan het predikaat 'genormaliseerd' toekennen.

In de volgende subparagraaf zullen we de rollen eens omkeren: producten krijgen daar de hoofdrol, terwijl de gerechten daaraan ondergeschikt worden gemaakt.

Omkering van herhalende groepen

Een gerecht kan meerdere producten als ingrediënt bevatten. Daarom is de samengestelde kolom 'ingrediënten' een herhalende groep, zie figuur 1.5. Omgekeerd kunnen we stellen: een product kan als ingrediënt voorkomen in meerdere gerechten. De consequentie is dat we 'gerechten' moeten kunnen weergeven als herhalende groep ten opzichte van producten. Het is niet moeilijk die omkering uit te voeren, zie figuur 1.9.

Product

naam	eenheid	energiePE	gerechten				
			gerecht	energiePP	bereidingstijd	bereidingswijze	hoeveelheidPP
ijs	liter	1600	Coupe Kiwano Glace Terrace	431 403	20 5	Schil ... Neem ...	0.15 0.2
kiwano	stuks	40	Coupe Kiwano	431	20	Schil ...	0.5
slagroom	deciliter	336	Coupe Kiwano	431	20	Schil ...	0.3
suiker	gram	4	Coupe Kiwano	431	20	Schil ...	10
tequila	eetlepel	30	Coupe Kiwano	431	20	Schil ...	1
aardbeien	gram	0.25	Glace Terrace Mango Plus Plus	403 131	5 8	Neem ... Snijd ...	50 50
pernod	eetlepel	35	Glace Terrace	403	5	Neem ...	2
peper			Glace Terrace	403	5	Neem ...	
mango	stuks	80	Mango Plus Plus	131	8	Snijd ...	0.5
zure room	deciliter	195	Mango Plus Plus	131	8	Snijd ...	0.4

Figuur 1.9 Gerechten als herhalende groep bij ingrediënten

De tabel heet nu Product: er is één rij per product. Er zijn vier kolommen, waarvan er één samengesteld is: per product is er een subtabelletje van gerechtgegevens. De subtabelletjes hebben vier kolommen met 'pure' gerechtinformatie. De vijfde kolom echter, hoeveelheidPP, bevat informatie die betrekking heeft op de *combinatie* van een product en een gerecht. Vandaar dat deze zowel in figuur 1.5 als in figuur 1.9 deel uitmaakt van de herhalende groep.
Merk op dat de tabelstructuur van Product, in tegenstelling tot die van Gerecht, toelaat om producten zonder gerechten op te nemen. Bij een product zonder gerecht wordt de gerechtencel leeg gelaten.

De omkering illustreert het punt 'gebrek aan symmetrie' van paragraaf 1.2. We kunnen het ook 'willekeur' noemen. Een weergave met herhalende groepen behandelt de 'dingen' (gerechten en producten) waarvan eigenschappen in de tabel voorkomen ongelijkwaardig. De kracht van een goed model is juist gelegen in een gelijkwaardige behandeling, onafhankelijk van de manier waarop een gebruiker later, via een applicatie, de gegevens gepresenteerd wil zien. Ook moet het mogelijk zijn om later geheel nieuwe toepassingen bij dezelfde database te realiseren.

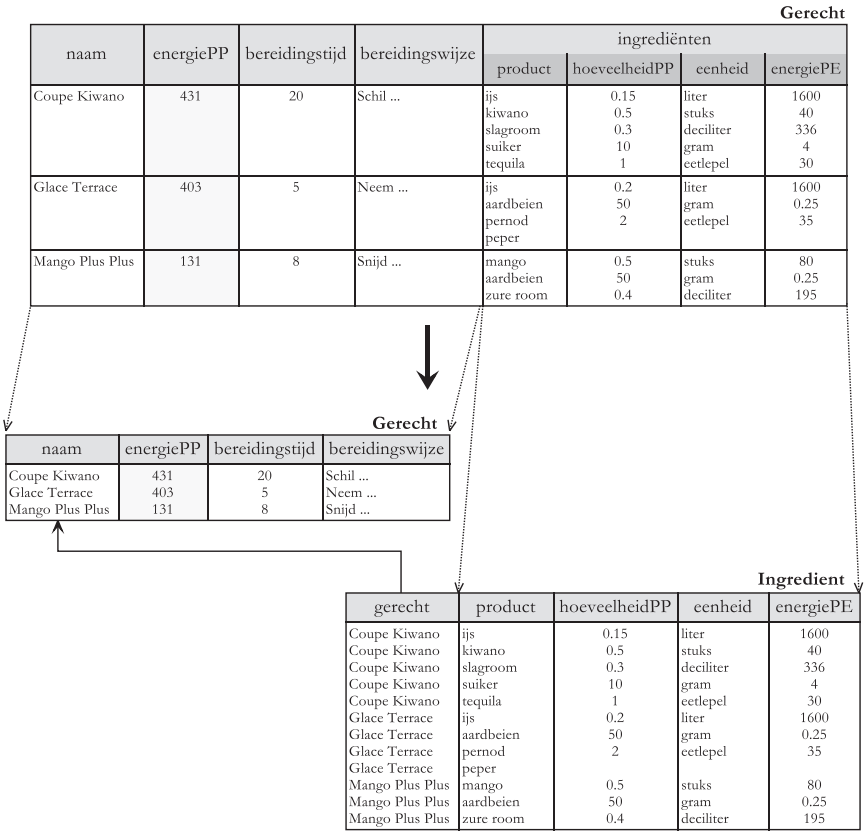
In de volgende paragraaf zullen we transformaties uitvoeren waarbij we de herhalende groep en de redundantie kwijtraken.

1.5 Verbeterde tabelstructuren

Herhalende groepen en redundantie komen voort uit het feit dat er informatie over meerdere 'soorten dingen' in één tabel zit. In de volgende paragrafen worden zowel de herhalende groep als de gesignaleerde redundantie geëlimineerd door een bepaalde vorm van *tabelsplitsing* (of *decompositie*) toe te passen. Het uiteindelijke resultaat voldoet aan het principe 'elk soort ding zijn eigen tabel'. Van nu af zullen we de 'dingen' in de werkelijkheid (concreet of abstract) aanduiden met *entiteiten* en 'soorten van dingen' met *entiteittypen*. We sluiten daarbij aan bij veelgebruikte terminologie uit de theorie en de praktijk van het informatiemodelleren. In deze terminologie luidt het principe: 'elk *entiteittype* zijn eigen tabel'.

Elimineren van de herhalende groep

Om de herhalende groep in de tabel van figuur 1.5 kwijt te raken, is een eenvoudige ingreep voldoende. Deze komt er ongeveer op neer dat we de herhalende groep er 'afknippen', zie figuur 1.10. Zouden we het bij knippen laten, dan zou er informatie verloren gaan: hoe weten we dan nog bijvoorbeeld dat de bovenste rij (ijs; 0.15; liter; 1600) bij het gerecht Coupe Kiwano hoort? Vandaar dat nog een kolom is toegevoegd voor het bijbehorende gerecht.



Figuur 1.10 Structuurtransformatie: elimineren herhalende groep

De resultaatstructuur omvat twee tabellen zonder herhalende groep. Elke cel bevat een enkelvoudige waarde; er zijn geen cellen meer met een subtabel als waarde. De afgesplitste tabel heeft de passende naam Ingredient gekregen. De waarden in de toegevoegde kolom gerecht van Ingredient verwijzen naar de waarden in de kolom naam van Gerecht.
De redundantie is echter door de nieuwe structuur niet geëlimineerd. Die moeten we dus nog zien kwijt te raken.

DIT BOEK BEHANDELT DE RELATIONELE DATABASETHEORIE EN DE RELATIONELE DATABASETAAL SQL AAN DE HAND VAN AANSPREKENDE VOORBEELDEN EN HELDER UITGELEGDE THEORIE. DE BENADERING IS INFORMEEL EN PRAKTISCH, MAAR TEGELIJKERTIJD CONCEPTUEEL. DOOR DE DIDACTISCHE BENADERING WORDT MET RELATIEF WEINIG MOEITE EEN HOOG NIVEAU BEREIKT.

Het boek omvat de volgende delen:

- Deel A behandelt de componenten van een relationeel systeem: de 'achterkant' (structuur en regels van een relationele database), de 'voorkant' (de structuur van applicaties, in relatie met die van een database) en de communicatie tussen beide via SQL. Dit deel sluit af met een informele, moderne behandeling van de normalisatietheorie.
- De delen B en C bevatten een praktische en grondige cursus SQL, in voortdurende relatie met de theorie.
- Deel D is verdiepend, met hoofdstukken over het methodisch aanpakken van queryproblemen, query-optimalisatie, het implementeren van databaseregels via triggers en stored procedures en metagegevens in de data dictionary.

In de vierde druk wordt de laatste versie van Firebird 3.0 als databasemanagementsysteem gebruikt, een open source product waarvan de SQL nauw aansluit bij de standaard. Hierdoor kan het datatype boolean geïntroduceerd worden. Ook zijn tal van verbeteringen aangebracht in de tekst, waarbij onder meer het oude hoofdstuk Autorisatie is verbreed vanuit de modernere invalshoek Security.

Op www.relationeledatabasesensql.nl is aanvullend en ondersteunend materiaal te vinden, zoals uitwerkingen van de opgaven, scripts en diagrammen bij alle voorbeelddatabases en extra teksten. Daarnaast is er de 'Boekverkenner' te downloaden, een krachtige en gebruikersvriendelijke SQL-omgeving, geïntegreerd in een interactieve leeromgeving. Ook is hier het online-boek te raadplegen.

Dit boek is geschikt voor databasecursussen binnen het hoger onderwijs en beginnende en gevorderde SQL-programmeurs in de praktijk.

LEO WIEGERINK heeft veel ervaring met lesgeven en cursusontwikkeling aan de Hogeschool van Amsterdam en de Open Universiteit.

JEANOT BIJPOST en **MARCO DE GROOT** houden zich bezig met de ontwikkeling en het beheer van grote informatiesystemen.

www.relationeledatabasesensql.nl
www.boomhogeronderwijs.nl

9 789024 429936 >