
Boom
Database modelleren
Peter ter Braake

Database modelleren

Peter ter Braake

Boom

Opmaak binnenwerk: Holland Graphics, Amsterdam
Basisontwerp omslag: Dog & Pony, Amsterdam
Omslagontwerp: Coco Bookmedia, Amersfoort
Beeld omslag: Liu zishan, Shutterstock

ISBN 9789024429554
ISBN 9789024429561 (e-book)
NUR 123

www.boomhogeronderwijs.nl

Woord vooraf

Voor je ligt het boek *Database modelleren*. Data is tegenwoordig belangrijker dan ooit. Databases zijn daarmee ook steeds belangrijker. Databases worden voor veel verschillende doeleinden gebruikt. Achter elke webshop zit een database. De dokter gebruikt een database om patiëntgegevens bij te houden en een callcenter gebruikt een database om elk gesprek te registreren.
Ook verzamelen we gegevens om deze vervolgens te kunnen analyseren en baseren we rapporten op de resultaten. Nog altijd worden veelal aparte rapportagedatabases of datawarehouses gebouwd.
Met het groeien van de verscheidenheid aan gegevens en de steeds geavanceerdere dingen die we met de gegevens doen, zijn ook de verschillende soorten databases gegroeid. Naast relationele databases kom je steeds vaker zogenaamde noSQL-databases tegen.
Hoe verschillend al deze databases ook zijn, ze hebben één ding gemeen. Het succesvol inzetten van een database begint met het ontwerpen van de juiste structuur. Dat kan de tabelstructuur van een relationele database zijn, maar het kan ook over de partitionering van een noSQL-cluster gaan. Vaak krijg je maar één kans om een database te ontwerpen. Dat ontwerp bepaalt in hoge mate het succes van de database.
Dit boek gaat in grote mate over het ontwerpen van relationele databases, dus over het bepalen welke tabellen je moet maken om een goede database op te zetten. We gaan in op de drie grote richtingen: normaliseren, dimensioneel modelleren en Data Vault. Relationele databases vormen nog altijd het overgrote deel van de in gebruik zijnde databases. In hoofdstuk 6 komen in wat algemenere termen noSQL-databases aan bod.

Voor wie is dit boek

Dit boek richt zich voornamelijk op het ontwerpen van databases. Hoe zet je een database goed op? Een database wordt pas zinvol in een groter geheel. Mensen moeten gebruik maken van de gegevens die in de database zijn opgeslagen. Deze mensen gebruiken applicaties om dat te doen. Dat kunnen selfservice Business Intelligence-tools zijn en het kan ook zijn dat je als klant ingelogd bent in een webshop om iets te kopen.
In dit boek gaan we ervan uit dat een data engineer zich in brede zin richt op het opslaan en gebruik van gegevens binnen een organisatie. Hij of zij is betrokken bij het maken van keuzes met betrekking tot welke database het meest geschikt is. De data engineer houdt zich ook bezig met de overall data-architectuur. Hij bepaalt (mede) wie welke gegevens in welke vorm krijgt aangeboden. En de data engineer zet de database op en maakt het ontwerp. Dat is het deel waar dit boek zich op richt.
Gezien dit laatste had het boek wellicht beter *Database Design voor de database-ontwikkelaar* kunnen heten. Maar in het huidige tijdperk van DevOps en alle snelle ontwikkelingen rond de cloud en Big Data is dat een te beperkte en ouderwetse term geworden. Dit boek is voor iedereen die betrokken is bij het opzetten en inrichten van een database, relationeel of anders. Vandaar de titel *Database modelleren*, zonder een toevoeging

Voorkennis

Dit boek gaat er niet van uit dat je een data engineer bent. Ook is voorkennis van databases niet noodzakelijk. We beginnen bij het begin: wat is een database eigenlijk en waarom gebruiken wij databases? Vanaf dat startpunt word je meegenomen tot op het punt dat je zelf een database-ontwerp kunt maken.
Voorkennis van de taal SQL is een pre, maar geen noodzaak. Nadat je dit boek gelezen hebt, zal het makkelijker zijn SQL te leren. Andersom is dit zeker ook waar. Met enige kennis van SQL zullen de concepten uit dit boek makkelijker te begrijpen zijn.
In het boek komen enkele data-architectuurplaatjes en -overwegingen aan bod. Enige voorkennis van wat Business Intelligence is en welke rol datawarehouses daarin spelen is een pre, maar wederom geen noodzaak. Deze kennis kun je eventueel opdoen met het boek *Leerboek Business Intelligence* (ter Braake, 2018).

Inhoudsopgave

1

Inleiding databases 1

Welkom in het boek *Database modelleren*. Data (gegevens) is de afgelopen jaren steeds belangrijker geworden. Bijna alle applicaties gebruiken gegevens, of die applicatie nu een Customer Relationship Management (CRM-)applicatie op het werk is, of een socialmedia-app op je (privé)telefoon. Al die gegevens worden opgeslagen in databases. Vanaf de jaren tachtig van de vorige eeuw tot vrij recentelijk waren dat (bijna allemaal) relationele databases. Tegenwoordig zien we steeds vaker noSQL-databases (noSQL staat voor 'not only' SQL). SQL, ofwel Structured Query Language, is de programmeertaal achter alle relationele databases.

noSQL staat voor 'not only' SQL

De term noSQL refereert daarmee aan het feit dat het opslaan van gegevens niet langer het domein is van alleen de relationele databases, maar dat er ook andere soorten databases bestaan. Die andere databasesoorten vormen het terrein van de Big Data.

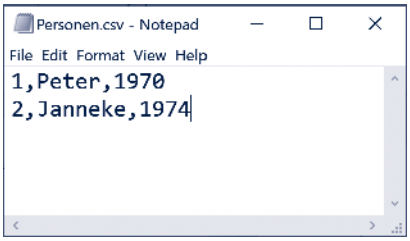
Dit boek volgt het historisch verloop van databases, te beginnen in de jaren tachtig van de vorige eeuw. Hoofdstuk 1 gaat over het ontstaan van de relationele database in die tijd en gaat uitgebreid in op wat een relationele database eigenlijk is. In hoofdstuk 2 (Entiteitenanalyse) en hoofdstuk 3 (Normaliseren) gaan we uitgebreid in op hoe een relationele database ontworpen moet worden. In hoofdstuk 4 gaan we in op datawarehouses. In de jaren negentig kwam men erachter dat het normaliseren, zoals oorspronkelijk voorgesteld bij de introductie van relationele databases, niet altijd ideaal bleek te zijn. Dimensioneel modelleren werd als aanvulling op normaliseren geïntroduceerd en komt in hoofdstuk 4 aan bod.

In hoofdstuk 5 komt modelleren volgens Data Vault aan bod. Deze modelleertechniek is ontwikkeld toen bleek dat normaliseren en dimensioneel modelleren voor meer historische opslag allebei nog tekortkomingen met zich meebrachten. In hoofdstuk 6 leer je meer over Big Data en noSQL-databases. Dit zijn andere soorten databases, die naast relationele databases zijn ontstaan in het tijdperk van de Big Data. In hoofdstuk 7 wordt beschreven welke laatste overwegingen er

nog zijn om het gemaakte datamodel echt te gaan maken. De echte implementatie hangt af van het platform dat je gaat gebruiken, maar je moet in alle gevallen het datamodel nog vertalen naar een fysieke implementatie.

1.1 Het ontstaan van relationele databases

Toen mensen net applicaties gingen programmeren op computers bestonden er nog geen databases. Gegevens die moesten worden bewaard, werden opgeslagen in bestanden. Vaak waren dat 'simpele' Comma Separated Value files, ofwel CSV-bestanden. Die zien er bijvoorbeeld uit zoals in figuur 1.1.



Figuur 1.1 Personen.csv

1.1.1 Bestanden

In de header van het screenshot van figuur 1.1 kun je nog zien dat het bestand Personen.csv heet. Er zullen dus wel gegevens over personen in dit bestand zitten. Of dat patiëntgegevens, klantgegevens, medewerkergegevens of andere gegevens zijn, valt niet te achterhalen met alleen het bestand.
Het gebruik van bestanden (ook wel platte bestanden of flat files genoemd) om gegevens op te slaan, brengt drie problemen met zich mee:

- Je kunt uit het bestand niet opmaken waar de gegevens over gaan.
- Het is niet flexibel en slecht voor de snelheid van gegevensverwerking.
- Het is erg lastig om met meer gebruikers tegelijkertijd te werken met dezelfde gegevens.

Je kunt uit het bestand niet opmaken waar de gegevens over gaan
Je kunt zien dat elke regel in het bestand twee komma's bevat en daarmee drie kolommen heeft. De tweede kolom is waarschijnlijk een voornaam. Een redelijke aanname gebaseerd op kennis van het Nederlands. In het Chinees is dit minder evident. De derde kolom is ook voor Nederlanders niet te gokken. Het zou om een jaartal kunnen gaan (geboortejaar?), maar het zouden net zo goed de vier cijfers van een postcode kunnen zijn, of een maandsalaris.
Het probleem dat hier wordt beschreven, is dat het bestand alleen gegevens bevat en dat de metadata ontbreken. Metadata zijn de gegevens die de gegevens beschrijven. Met kolomnamen zoals bijvoorbeeld Patiëntennummer, Pa-

tiënt_voornaam, Postcode wordt het al duidelijker. Die kolomnamen worden dan ook vaak als eerste regel toegevoegd.

Metadata zijn de gegevens die de gegevens beschrijven.

Maar er valt meer te weten over deze gegevens. Postcodes kun je niet optellen. Hoewel in dit geval alleen cijfers gebruikt worden en geen letters, is het eigenlijk een alfanumerieke 'code' waar je niet mee kunt en wilt rekenen. Als de laatste kolom een salaris was geweest had je er misschien wel mee moeten rekenen, bijvoorbeeld om een jaarsalaris te berekenen uit het maandsalaris. De kolom zou in dat geval een numerieke waarde bevatten. Met andere woorden: we willen het gegevenstype (datatype) weten van de gegevens.

Bij gegevens opgeslagen in bestanden zijn de gegevens en de metadata gescheiden.

Het is niet flexibel en slecht voor de snelheid van gegevensverwerking

Nog vervelender wordt het als we vanuit programmacode met dit bestand gaan werken. Stel, je wilt de postcode weten van Janneke. Je programmeert dan waarschijnlijk iets als:

1. lees een regel;
2. lees de tweede kolom;
3. als de waarde gelijk is aan Janneke geef dan de derde kolom terug;
4. herhaal bovenstaande totdat er geen regels meer zijn.

Met twee regels in dit bestand heb je snel resultaten. De performance zal bij grotere bestanden echter al snel traag worden.
Erger nog is als iemand het bestand een beetje aanpast. Tussen de tweede en de derde kolom voegen we een nieuwe kolom in met de achternaam van de patiënt. De bovenbeschreven code werkt niet meer omdat ervan uit wordt gegaan dat we de derde kolom willen weten. Dat is nu de achternaam en niet langer de postcode. De vraag om postcode zou onafhankelijk moeten zijn van welke kolom de postcode bevat.

Het is erg lastig om met meer gebruikers tegelijkertijd te werken met dezelfde gegevens

In moderne applicaties hebben we niet te maken met een enkele gebruiker. Er werken meer mensen tegelijkertijd samen met dezelfde gegevens. Een database moet dat ondersteunen, 'gewone' bestanden ondersteunen dat niet. Voor relationele databases is zelfs beschreven waar een database aan moet voldoen om meer gebruikers tegelijkertijd te ondersteunen. Dat worden de ACID properties genoemd. In hoofdstuk 6 lees je daar meer over.
In eerste instantie zijn er allemaal slimme oplossingen bedacht om het werken met bestanden beter en efficiënter te maken. Bestandsformaten zoals ISAM files en VSAM files werden uitgevonden. Het gaat voor dit boek te ver om daarop in te gaan. Genoeg is te vermelden dat de scheiding van data en metadata tot

inflexibiliteit leidt en tot slechte performance. Vandaar dat databases zijn ontstaan.

1.1.2 Relationele databases

Een database (gegevensbank) is een zichzelf beschrijvende verzameling van bij elkaar horende gegevens die tot doel heeft mensen of machines van informatie te voorzien.

De eerste databases ontstonden in de jaren zestig van de vorige eeuw. Deze eerste systemen waren zogenoemde hiërarchische databases, iets later gevolgd door netwerkdatabases. Beide systemen boden echter nog niet de flexibiliteit die nodig is om in complexere organisaties met grote hoeveelheden gegevens te werken.
In het begin van de jaren zeventig van de vorige eeuw bedacht E.F. Codd, een Engelse wiskundige die voor IBM werkte, de theorie van het relationele databasemodel. Hij ging uit van de wiskundige verzamelingenleer. Deze wiskundige discipline beschrijft in enkele eenvoudige regels grote verzamelingen. Codd bedacht dat die regels ook van toepassing konden zijn op bijvoorbeeld de verzameling van alle klanten van een bedrijf. De regels zijn dus van toepassing op het soort informatie dat wij willen opslaan in databases.
De term 'relationele' database verwijst, in tegenstelling tot wat je vaak leest, naar het feit dat gegevens worden opgeslagen in tabellen. Neem bijvoorbeeld de verzameling getallen {1, 2, 3, 4}. Bedenk daarnaast een tweede verzameling, bijvoorbeeld de volgende namen: {Peter, Janneke, Jari, Mats}. Deze beide verzamelingen zou je kunnen combineren tot de tabel van figuur 1.2.

PatientID (int)	PatientNaam (nvarchar(50))
1	Peter
2	Janneke
3	Jari
4	Mats

Figuur 1.2 Patiëntentabel

Waar we begonnen met twee onafhankelijke verzamelingen, hebben we met de tabel een relatie gemaakt tussen deze beide verzamelingen. Door er een tabel van te maken, zeggen we dat 1 bij Peter hoort en 2 bij Janneke. Patiënt ID en Patiënt Naam zijn niet langer onafhankelijke verzamelingen maar vormen een relatie. Anders gezegd: de tabel is de relatie tussen Patiënt ID en Patiënt Naam. Algemener: in een relationele database worden gegevens opgeslagen in tabellen. "Relation" is in de theorie van Codd een ander woord voor "table".

Relationele databases hebben als kenmerk dat gegevens worden opgeslagen in de vorm van tabellen.

Figuur 1.2 laat nog iets extra's zien. Kolom 1 heeft als naam PatientID. Daaronder staat dat deze kolom gegevens bevat van het gegevenstype (datatype) int. Dat betekent dat de kolom gehele getallen bevat. Net zo bevat de tweede kolom alfanumerieke gegevens. Dat betekent zoveel als dat je letters, leestekens, speciale tekens, ofwel karakters kunt opslaan. Deze metadata zijn een onderdeel van de tabel. De gegevens en de metagegevens vormen één geheel.

In een database vormen gegevens en metagegevens één geheel.

Een relationele database bestaat over het algemeen niet uit slechts één tabel. In de praktijk kan een database zelfs uit duizenden tabellen bestaan. Volgens de definitie is een database een verzameling bij elkaar horende gegevens. De tabellen zullen dus onderlinge verbanden hebben. In het Engels zijn dat relationships, in het Nederlands relaties. En dat lijkt zo op 'relationeel', dat vaak gedacht wordt dat hier de naam relationele database vandaan komt.
Een RDBMS, ofwel Relational Database Management System, is een softwarepakket waarin je databases kunt aanmaken en beheren die gemaakt zijn volgens Codds regels van de relationele database. Een RDBMS is dus een product/applicatie waarin je tabellen kunt maken die je vervolgens kunt vullen met gegevens. Bovendien leveren deze applicaties allemaal extra diensten, zoals de mogelijkheid om de beveiliging van de gegevens zo in te regelen dat alleen de juiste mensen de gegevens mogen zien en bewerken. Vandaar het woord management. Het systeem stelt je in staat alles met je gegevens te doen, van gebruik tot en met beheer. Voorbeelden van RDBMS'en zijn Microsoft SQL Server, Oracle, IBM DB2, MySQL en Maria DB.

Een RDBMS is een databaseproduct dat werkt volgens de regels van Codds relationele model en je in staat stelt met gegevens te werken en die gegevens te beheren.

De vraag die nog beantwoord moet worden, is hoe deze relationele databases de flexibiliteit en performance geven die CSV-bestanden niet hadden.

1.2 Structured Query Language

Structured Query Language, SQL, is de taal die hoort bij relationele databases. Elke relationele database gebruikt SQL als de taal waarmee je met de database communiceert. Dat kan zijn het werken met de gegevens zelf. Maar ook het maken van databases zelf, het maken van tabellen in de database en het beveiligen van de gegevens doe je met SQL. SQL bestaat uit drie onderdelen:

- DCL – Data Control Language
- DDL – Data Definition Language
- DML – Data Manipulation Language

Data(bases) spelen een cruciale rol in de huidige digitale informatie-wereld. We verzamelen data om deze vervolgens te kunnen analyseren. Het succesvol inzetten van een database begint met het ontwerpen van de juiste structuur. Vaak krijg je maar één kans om een database te ontwerpen. Dat ontwerp bepaalt in hoge mate het succes.

Met het groeien van de verscheidenheid aan gegevens en de steeds geavanceerdere bewerking van deze gegevens, zijn ook de verschillende soorten databases gegroeid. Relationele databases vormen het overgrote deel van de databases die in gebruik zijn. Daarnaast kom je steeds vaker noSQL-databases tegen.

In *Database modelleren* leer je welk soort database – relationeel of SQL – geschikt is voor het beoogde doel en hoe je de structuur van een database kunt opzetten. Welke tabellen heb je nodig in een relationele database en wat voor partitionering heb je nodig voor je noSQL database? Voor relationele databases wordt het ontwerp nog vertaald naar een fysiek gegevensmodel met aandacht voor onder andere het kiezen van de juiste gegevenstypes.

Dit boek is voor iedereen die betrokken is bij het opzetten en inrichten van een database, relationeel of anders. We beginnen bij het begin: wat zijn databases en waarom gebruiken we ze? Vanaf dat startpunt leer je hoe je zelf een database-ontwerp kunt maken.

www.boomhogeronderwijs.nl

9 789024 429554