Databases

Access database

Geschreven door

Arjan Kamberg
https://www.linkedin.com/in/arjankamberg/

Leerjaar: 2022/ 2023

Versie Datum: 15-05-2023

Inhoudsopgave

Voorkennis

Voor deze cursus is niet een grote voorkennis nodig. Heel veel dingen worden voor het eerst of opnieuw uitgelegd. Wel worden begrippen minder uitvoerig uitgelegd dan in voorgaande cursussen.

We gaan databases uitleggen met de Access database. Dit is een simpele database waar alle aspecten voor een beginnerscursus inzitten. De interface die bij access zit maakt het ook, voor een database, toegankelijk. Jammer genoeg kan Access alleen op Windows gebruikt worden.

Access is te vinden in het Office pakket waar je met je school-licentie toegang op hebt.

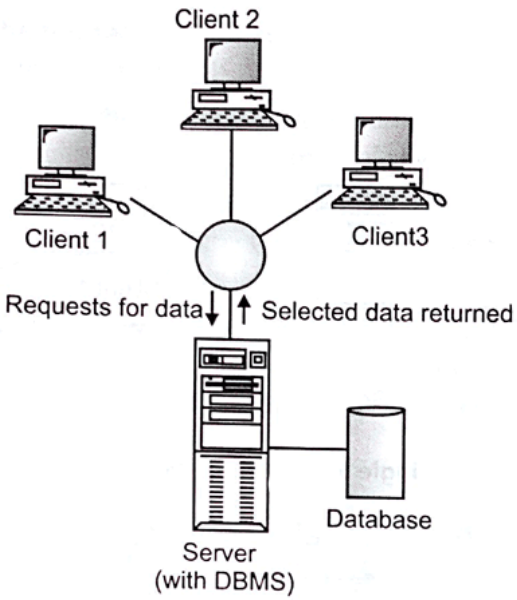
In dit cursusmateriaal wordt gebruikt gemaakt van Access 365 samen met Windows 10.

1 Introductie database

1.1 Wat is een database

Een database is een verzameling van gestructureerde informatie of data. Deze informatie wordt beheerd door een database beheersysteem oftewel een Database-Management System (DBMS). De data en de DBMS kan je weer gebruiken in databasesystemen.

In dit hoofdstuk gaan we een DBMS installeren op onze computer. In de DBMS gaan we leren hoe we gestructureerd data kunnen opslaan, en deze gaan we vullen met data en weer uitlezen.



Client server

We gaan in dit hoofdstuk alleen werken met relationele databases. Dit zijn databases waar van tevoren de relaties tussen velden gedefinieerd zijn. Dit is de meest efficiënte en snelste manier om data te benaderen. Maar het nadeel is dat deze databases weer minder flexibel zijn voor wijzigingen.

Er zijn verschillende aanbieders van databases, sommige zijn betaalde producten, anderen zijn gratis. Een gratis versie kan heel interessant lijken, maar kunnen ook weer als nadeel hebben dat ze minder updates krijgen of minder support opgeleverd wordt. Een paar hele populaire databases zijn:

- MySQL
- Microsoft Access
- Microsoft SQL Server
- FileMaker Pro
- Oracle Database
- dBASE
- PostgreSQL
- MOngoDB,
- MariaDB
- DB2

In dit boek gaan we werken met *Microsoft Access* en later met *MySQL*.

Microsoft Access is onderdeel van Microsoft office. Het is heel makkelijk te gebruiken en voornamelijk bedoeld voor enkele gebruikers.

MySQL is een gratis database dat op veel platformen werkt Het wordt gebruikt door een aantal van de grootste websites ter wereld zoals Airbnb, Uber, LinkedIn, Facebook, Twitter en YouTube.

Een database kan op je eigen computer staan, op een server of in de Cloud. Een database hoeft niet 1 enkele plek te hebben. Als een database in de Cloud staat wordt er meestal mee bedoeld dat tegelijkertijd de database op verschillende servers staat, en dat die versies met elkaar in verbinding staat. Het kan zijn dat een enkele database op tientallen plekken in de wereld staat. Als er een het niet doet, wordt je data, zonder dat je dat doorhebt, van een andere plek binnengehaald. Het kan ook zo zijn dat de databases zo groot zijn dat de data zelf in stukjes overal staan. Als je dan iets opvraagt uit de database wordt de informatie stukje voor stukje vanuit de hele wereld bij elkaar gezocht en naar je gestuurd.

Een Access database is een bestand op je computer met de extensie .accdb. Je lees dit bestand niet uit, je benader dit bestand. Als je het bestand opent dan vraag je aan een tussenprogramma gegevens uit het bestand op. Dit tussenprogramma snapt de structuur van het Access bestand en weet hoe het heel snel informatie eruit kan halen.

Om gegevens uit een database te halen, of juist gegevens erin te zetten is een taal ontwikkeld namelijk SQL. SQL staat voor Structured Query Language oftewel een gestructureerde vraagtaal. We leggen later uit hoe de taal werkt.

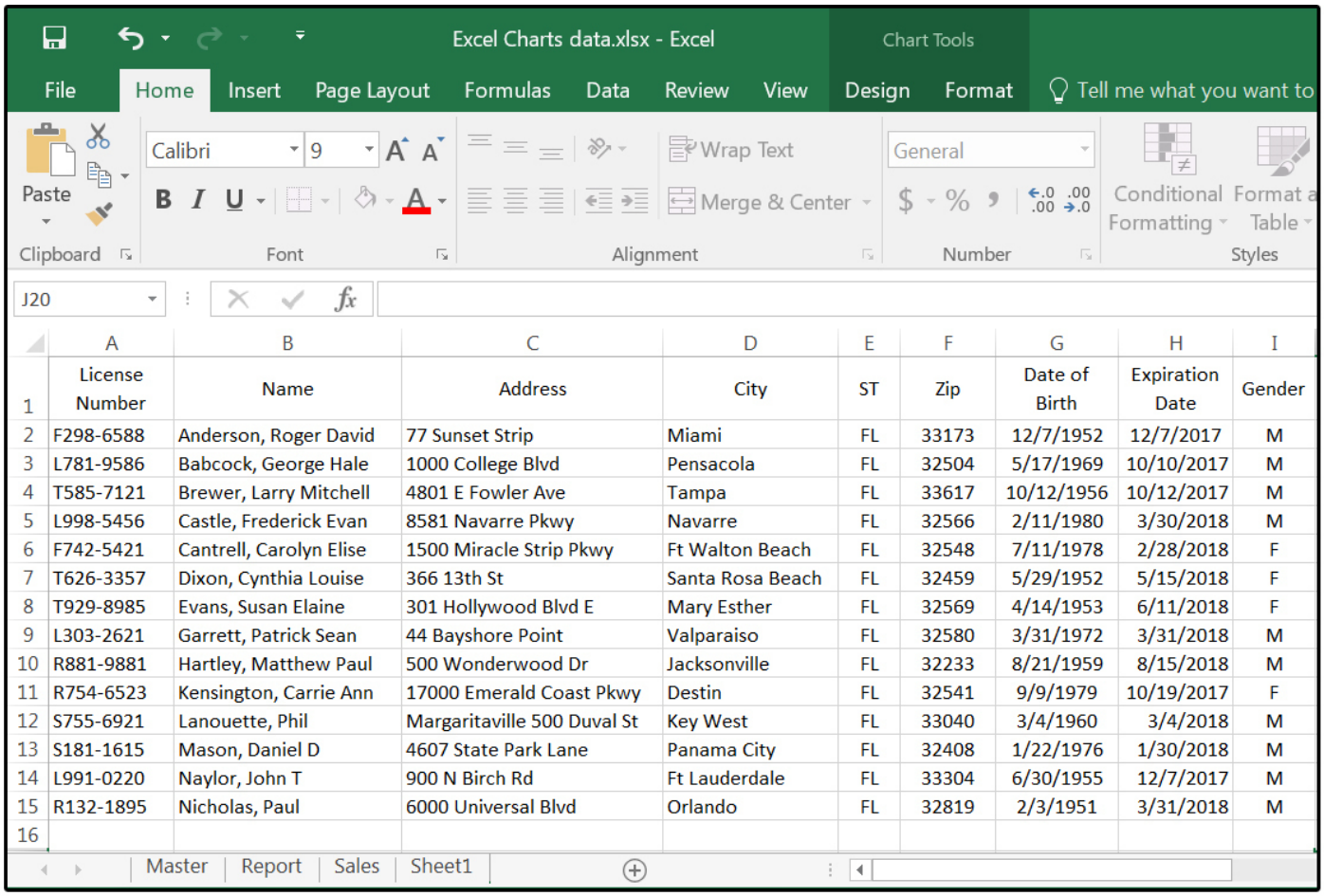
1.2 Verschillen met spreadsheet

Een database lijkt een op een spreadsheet (zoals Microsoft Excel of Apple Numbers). Beide kunnen ze informatie opslaan. Het grote verschil zit hem in de manier hoe data opgeslagen wordt en de hoeveelheid data dat opgeslagen kan worden. Ook kan je met een database veel beter regelen wie toegang heeft tot welke data. Spreadsheats zijn ontworpen voor enkele gebruikers die inzicht willen krijgen in een beperkte hoeveelheid data. Databases zijn ontworpen om grote hoeveelheden gestructureerde data voor meerdere gebruikers beschikbaar te stellen. Het filteren van de informatie kan je opvragen met complexe logica.

In een spreadsheet kunnen we een lijstje maken van personen en hun adres zoals hieronder. De kolommen hebben we een naam gegeven zoals License Number, Name, Address, City, Zip, Date Of Birth Zo kunnen we snel iemand opzoeken door in de lijst te gaan kijken.

De informatie die in iedere kolom staat is helemaal vrij om in te vullen. Plaatsnamen kunnen op verschillende manieren gespeld zijn, License Numbers hoeven zich niet aan regel te houden.

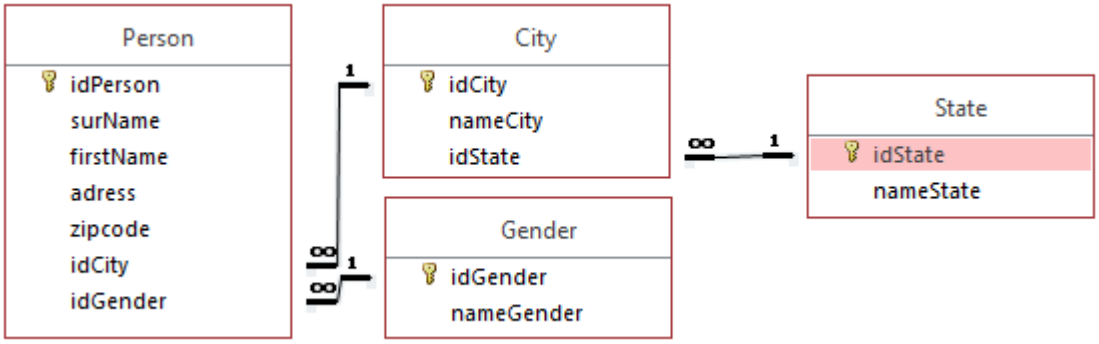
Ook al ziet de lijst van gegevens er netjes uit spreken we hier van ongeorganiseerde data. Als meerdere mensen in dezelfde stad wonen staat de naam van de stad er ook meerdere keren in. Dit kan met spelfouten zijn en het is ook nog heel veel extra opslagruimte om dat voor iedereen opnieuw op te slaan. Hier gaan we het nut inzien van databases.



	A	B	C	D	E	F	G	H	I
1	License Number	Name	Address	City	ST	Zip	Date of Birth	Expiration Date	Gender
2	F298-6588	Anderson, Roger David	77 Sunset Strip	Miami	FL	33173	12/7/1952	12/7/2017	M
3	L781-9586	Babcock, George Hale	1000 College Blvd	Pensacola	FL	32504	5/17/1969	10/10/2017	M
4	T585-7121	Brewer, Larry Mitchell	4801 E Fowler Ave	Tampa	FL	33617	10/12/1956	10/12/2017	M
5	L998-5456	Castle, Frederick Evan	8581 Navarre Pkwy	Navarre	FL	32566	2/11/1980	3/30/2018	M
6	F742-5421	Cantrell, Carolyn Elise	1500 Miracle Strip Pkwy	Ft Walton Beach	FL	32548	7/11/1978	2/28/2018	F
7	T626-3357	Dixon, Cynthia Louise	366 13th St	Santa Rosa Beach	FL	32459	5/29/1952	5/15/2018	F
8	T929-8985	Evans, Susan Elaine	301 Hollywood Blvd E	Mary Esther	FL	32569	4/14/1953	6/11/2018	F
9	L303-2621	Garrett, Patrick Sean	44 Bayshore Point	Valparaiso	FL	32580	3/31/1972	3/31/2018	M
10	R881-9881	Hartley, Matthew Paul	500 Wonderwood Dr	Jacksonville	FL	32233	8/21/1959	8/15/2018	M
11	R754-6523	Kensington, Carrie Ann	17000 Emerald Coast Pkwy	Destin	FL	32541	9/9/1979	10/19/2017	F
12	S755-6921	Lanouette, Phil	Margaritaville 500 Duval St	Key West	FL	33040	3/4/1960	3/4/2018	M
13	S181-1615	Mason, Daniel D	4607 State Park Lane	Panama City	FL	32408	1/22/1976	1/30/2018	M
14	L991-0220	Naylor, John T	900 N Birch Rd	Ft Lauderdale	FL	33304	6/30/1955	12/7/2017	M
15	R132-1895	Nicholas, Paul	6000 Universal Blvd	Orlando	FL	32819	2/3/1951	3/31/2018	M
16									

In een relationele database maken we een lijst met alle steden, en iedere stad krijgt een nummer. In een andere tabel met adressen typen we niet de naam van de stad in, maar het nummer. Zo hebben we maar een plek waar alle verschillende steden worden opgeslagen. Als er een spelfout is, hoef je dat maar een keer te wijzigen. Bij een kleine hoeveelheid adressen zal deze aanpak meer ruimte gebruiken, maar hoe meer adressen opgeslagen wordt, hoe minder ruimte het uiteindelijk zal gebruiken.

Een database zou op deze manier zijn opgebouwd. De gegevens zijn over verschillende tabellen onderverdeeld zodat er zo min mogelijk dubbele data wordt opgeslagen. Door middel van sleutels (getallen) wordt van de ene tabel naar een waarde in een andere tabel verwezen.



Figuur 1: Relatie diagram

1.3 Normaliseren

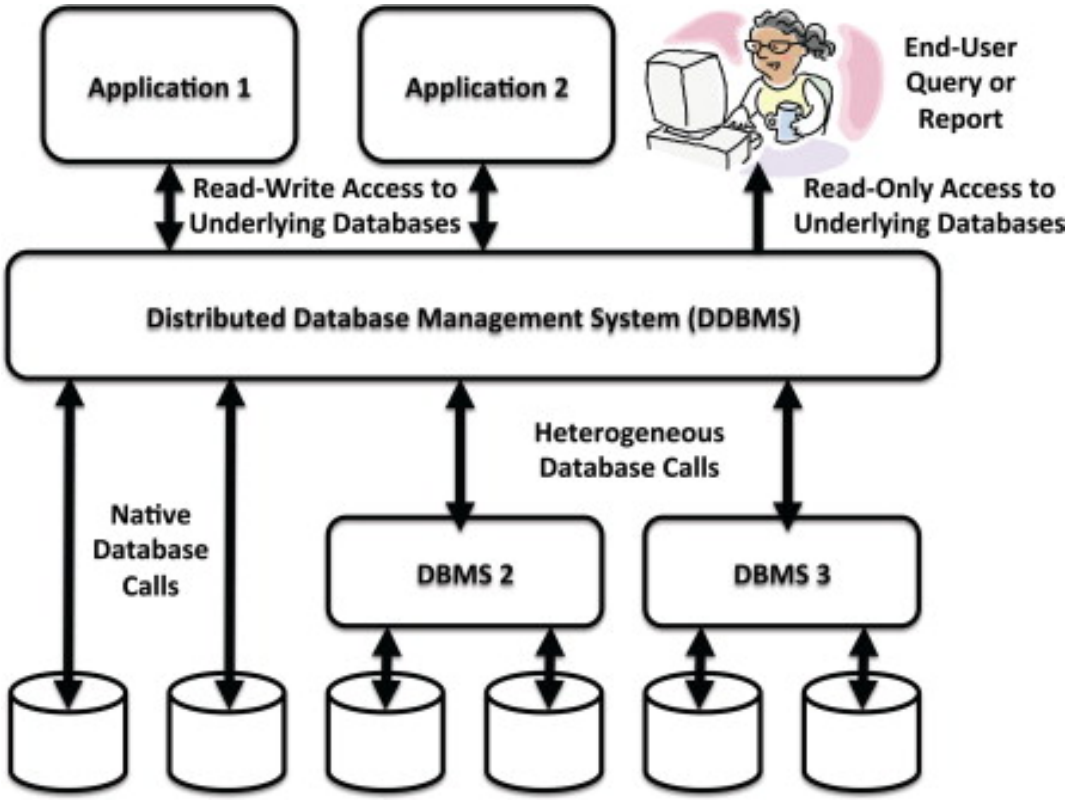
Het datamodel die we in “Figuur 1: Relatie diagram” zien bestaat uit een aantal kleine tabelletjes. Het opsplitsen van de data in allemaal kleine tabelletjes namen we in database-taal “Normaliseren”. Het voordeel van normaliseren is dat er heel veel herbruikbare structuren komen. We kunnen het gender van een persoon in een tekstveld opslaan, en daar kunnen we dan van alles in zetten omdat het een tekstveld is. Door normaliseren maken we een selectie van mogelijkheden. Hierdoor kan je meer sturing geven aan de waarden die ingevoerd kunnen worden. Bij het ontwerpen van een database kan je dan ook het beste zoveel mogelijk normaliseren.

1.4 Big data

Een database maken is alleen handig als je veel data wilt opslaan. Maar wat als je heel erg veel data wilt opslaan? Hiervoor gebruiken we nog steeds databases, maar hoe we die benaderen is anders.

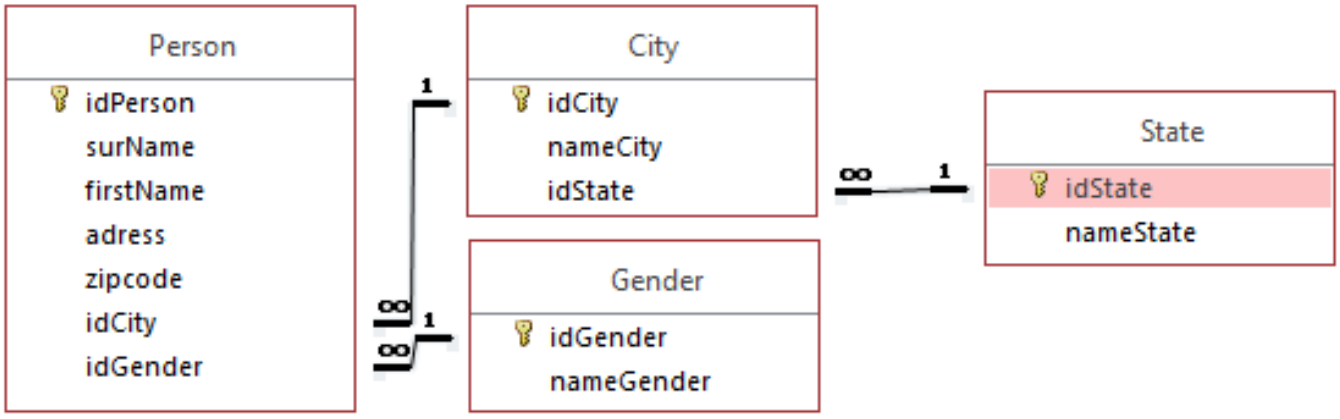
We zien dat alles tegenwoordig wel data genereerd. Je telefoon geeft continue door waar je bent, deze informatie wordt opgeslagen. Deze informatie wordt niet alleen van jou opgeslagen, maar van miljarden mensen, ieder uur van de dag weer. Gebouwen worden gemaakt met prefab-betonblokken met daar in ieder blokje een chip met de informatie wat voor materiaal het is, wanneer gemaakt en wat de afmetingen waren. Misschien ook met informatie waar het allemaal is gemaakt. Als een gebouw wordt afgebroken kunnen de blokken weer gebruikt worden voor een ander gebouw. Omdat alle blokken overal ter wereld zo opgeslagen zijn in een database, kan weer een nieuw gebouw gemaakt worden als in de database bekend is welke gebouwen weer uit elkaar gehaald worden, dan kan je ook weer een nieuw gebouw maken met beschikbare blokken. Dit zijn twee voorbeelden van duizenden projecten die zo bezig zijn, en alles wordt weer ergens op een database opgeslagen. Dit hoef ook niet allemaal dezelfde database te zijn, of zelfs dezelfde soort database.

Stel we willen alle plaatsnamen in de wereld. We kunnen deze opdracht geven aan een server die dit gaat uitzoeken in zijn eigen database. Het is alleen sneller als de taak wordt opgesplitst. We vragen aan 7 servers om ieder een continent te bekijken. Dan is het resultaat al 7x sneller gevonden. Nu kan de server voor ieder continent weer voor ieder land een zoekopdracht maken. Deze zoekopdrachten worden dan ook weer aan verschillende servers gestuurd. Zo kan uiteindelijk misschien wel 180 verschillende servers bezig zijn met jouw ene zoekopdracht. De server van Luxemburg zou snel klaar zijn. De Verenigde Staten van America is weer wat langer bezig, maar misschien heeft die de opdracht ook weer opgesplitst. Vanaf alle kanten komt het eindresultaat dan weer naar boven. Wat het eerste gevonden is komt als eerste binnen, en uiteindelijk tot de langzaamste server.



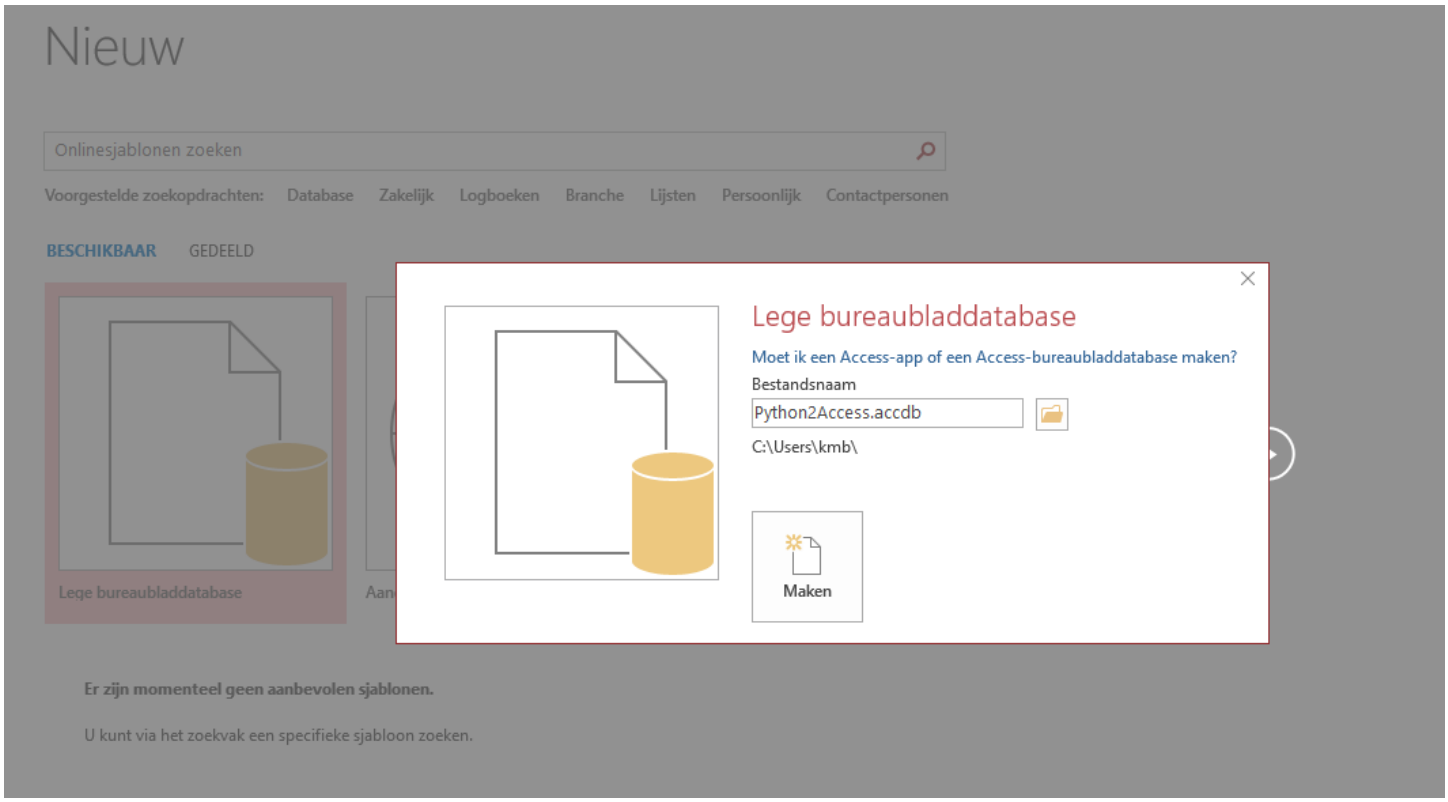
2 Microsoft Access Database maken

In “Figuur 2: Relatie diagram” zien we verschillende vakjes die tabellen voorstellen, en een lijntje ertussen die de relatie tussen de tabellen beschrijft. In dit hoofdstuk gaan we stap voor stap uitleggen hoe we deze database maken.



Figuur 2: Relatie diagram

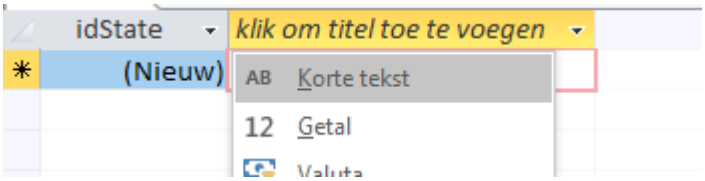
We beginnen met het aanmaken met de optie Lege bureaubladdatabase”. Daarna geven we de database een naam en we selecteren waar we hem willen opslaan. Als we op maken hebben gedrukt dan is de database gemaakt. We hebben nu een lege omgeving waar we tabellen in kunnen doen.



We beginnen als eerste met de tabel die het meest simpel is, en geen verwijzingen naar andere tabellen heeft. Dit betekent eigenlijk een tabel die het meest lijkt op een Excel tabel. In het voorbeeld “Figuur 2: Relatie diagram” is dat de tabel State, waar we de provincies in op gaan slaan.

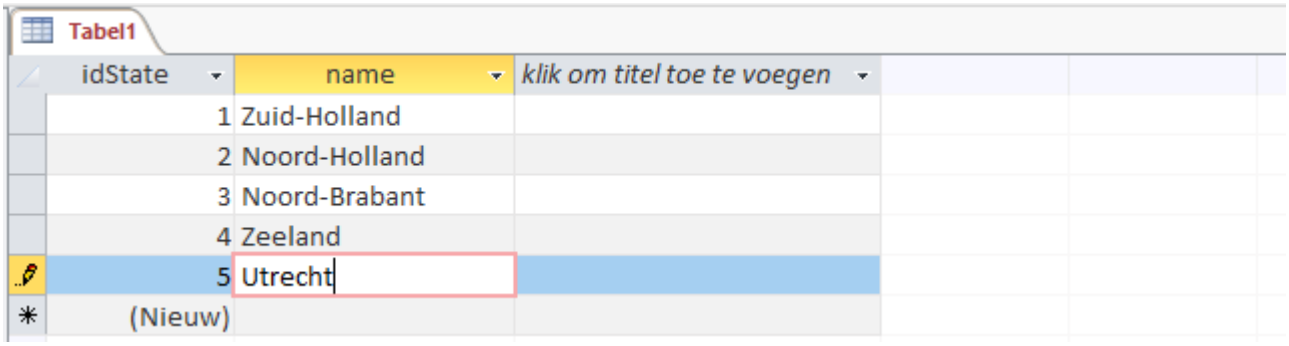
Standaard komt er al een lege tabel tevoorschijn met een veld genaamd id. Die naam wijzigen we direct naar idState (let weer op de schrijfwijze). De tabel gaan we later State noemen, de id geven we dan de naam id + {Tabelnaam}.

Op het veld ernaast drukken we op de rechtermuisknop, en daar maken we een “korte tekst”-veld aan met de naam “name” je kunt ook “nameState” gebruiken als je de waarschuwing van Access niet wilt zien. Het is belangrijk dat je bij al je tabellen dan consequent ben hoe je de benaming doet. Doordat we -State aan het einde van de naam zetten hebben we dezelfde structuur als we bij het id-veld hebben gebruikt. We beginnen de naam van een veld altijd met een kleine letter, en verder gaan we weer via camel case de volgende hoofdletters bepalen.

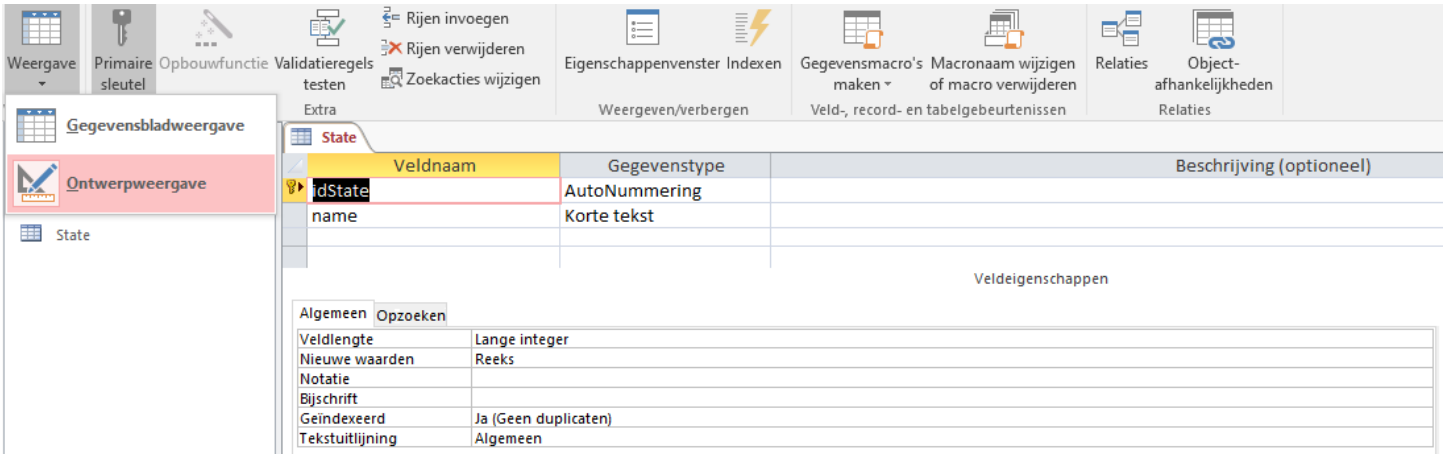


We kunnen nu al de namen van de provincies allemaal invoeren, zoals hieronder weergegeven. Iedere keer als je een nieuwe naam ingeeft, wordt er automatisch een nummer voorgezet. Dit nummer is de sleutel waarmee de provincie verder bekend is. Je kan het zien als je studenten-nummer waarmee jij als student bekend ben.

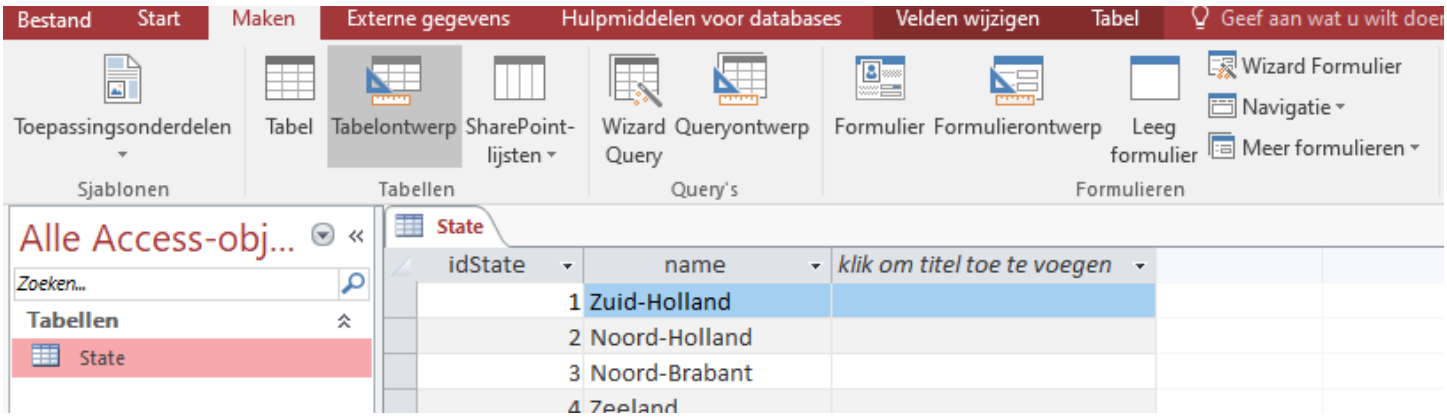
Je kunt nu <CTRL><S> drukken om de tabel op te slaan met een andere naam. We noemen de tabel State (met een hoofdletter S). Als je wijzigingen aanmaakt in de velden van een database hoef je niet iedere keer op te slaan, iedere wijziging wordt direct opgeslagen als je op <Enter> drukt.



Links bovenaan staat een knop met weergave. Bij deze knop kan je kiezen op wat voor manier de tabel weergegeven wordt. Dit was “Gegevensbladweergave” maar dat kunnen we nu wijzigen in “Ontwerpweergave”. Hier kunnen we zien hoe de kolommen heten en wat de eigenschappen zijn van de kolommen. We zien bij idState dat dit een Auto Nummering veld is, dat hebben we net ook kunnen zien bij het invoeren van nieuwe provincienamen.

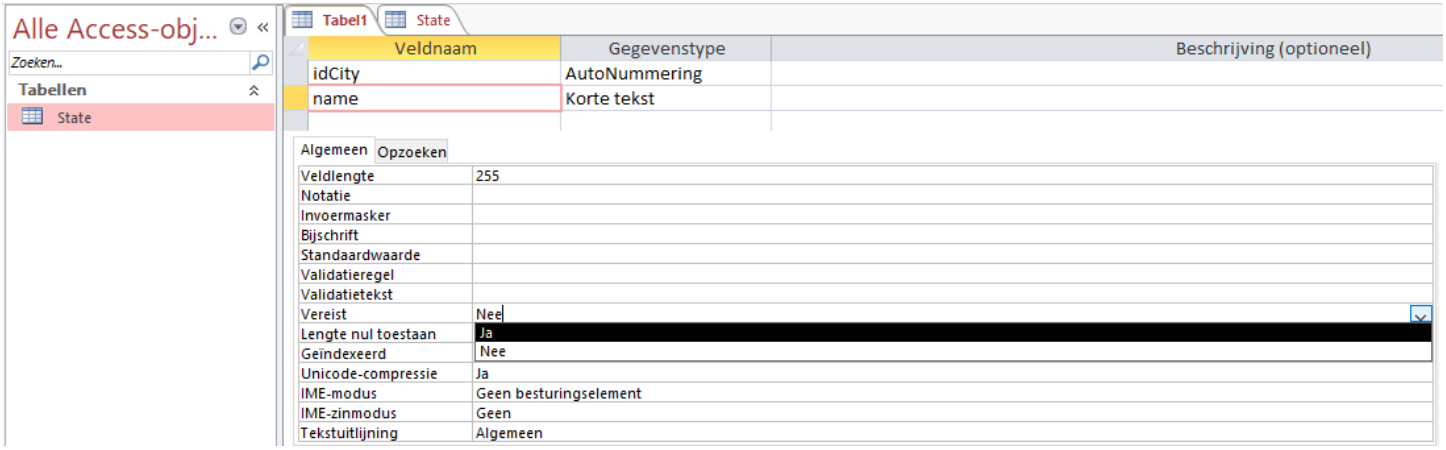


Als we naar Maken gaan in het hoofdmenu dan kunnen we een Tabel maken, of een Tabelontwerp. De State tabel hebben we gemaakt op de Tabel manier. Als we naar Tabelontwerp gaan dan zien we de Ontwerpweergave die we zojuist gezien hebben.



We gaan nu de City-tabel maken waar alle steden in komen te staan. Ook hier maken we als eerste veld een id-veld aan met de naam idCity, en bij Gegevenstype selecteren we “AutoNummering“.

De volgende regel noemen we name, en dat is een “Korte tekst“. We willen dat er altijd een naam wordt ingevoerd, dat kunnen we instellen door bij de opties “Vereist” op “Ja” te zetten. Als we nu de tabel willen opslaan onder de naam City (<CTRL><S>) dan krijgen we een melding dat we voor de tabel geen primaire sleutel hebben gedefinieerd. Als we op “Ja” drukken wordt het eerste veld als primaire sleutel ingesteld. We gaan nu eerst alle provincies in de tabel State zetten, en in de City tabel zetten we wat plaatsen.



- **Alle namen in een database zijn camel case geschreven.**
- **Kolom namen van een tabel beginnen altijd met een kleine letter.**
- **Tabel namen beginnen altijd met een hoofdletter.**
- **De namen van tabellen zijn altijd enkelvoudig (dus Person en niet Persons) ook al staan er meer personen in.**
- **De namen van de kolommen zijn altijd zo kort en krachtig mogelijk, en beschrijven het veld.**
- **Het identifier-veld is altijd in de vorm van id + {Tabel Naam}**

2.1 Primary-key en Foreign-key

We hebben bij beide tabellen een primary-key ingesteld. Dit is een uniek nummer waarmee de data die verder op de regel staat mee gevonden kan worden. Als je het nummer weet, kan je heel snel in de tabel zoeken wat erbij hoort. Zo ook met een studenten nummer, met het nummer kan je heel snel gegevens van iemand vinden. Als je alleen je achternaam noemt, kan het zijn dat er meerdere met dezelfde achternaam zijn. Zelfs voor en achternaam hoeft niet uniek te zijn, De primary-key is binnen de database die we maken per definitie uniek. De primary-key is ook geïndexeerd, dat wil zeggen dat er een slimme manier achter zit die heel snel aan de hand van het nummer de andere gegevens kan vinden. De database hoeft dan niet regel voor regel langs te gaan of het id op die regel hetzelfde is als de opgegeven id.

We hebben nu twee tabellen, een met provincies en de andere met plaatsen. Nu weten we dat alle plaatsen in een provincie moeten zitten. We kunnen dus in “Databaseontwerp” een veld achter de name plaatsen met daarin de naam van de provincie. Dit is wat we juist NIET willen in een relationele database. We willen een relatie maken met een andere tabel waar die informatie inzit. De manier om te verwijzen naar die andere tabel is met die unieke sleutel. We willen niet de naam van de provincie opslaan, maar het nummer, oftewel de primary-key, van die provincie. We maken een veld idState aan, Numeriek, aan en vullen de nummers in van de provincie die bij iedere plaats hoort. De waarden van idState in de City tabel zijn niet uniek zoals je kunt zien, maar ze verwijzen allemaal naar een waarde in de State tabel.

State

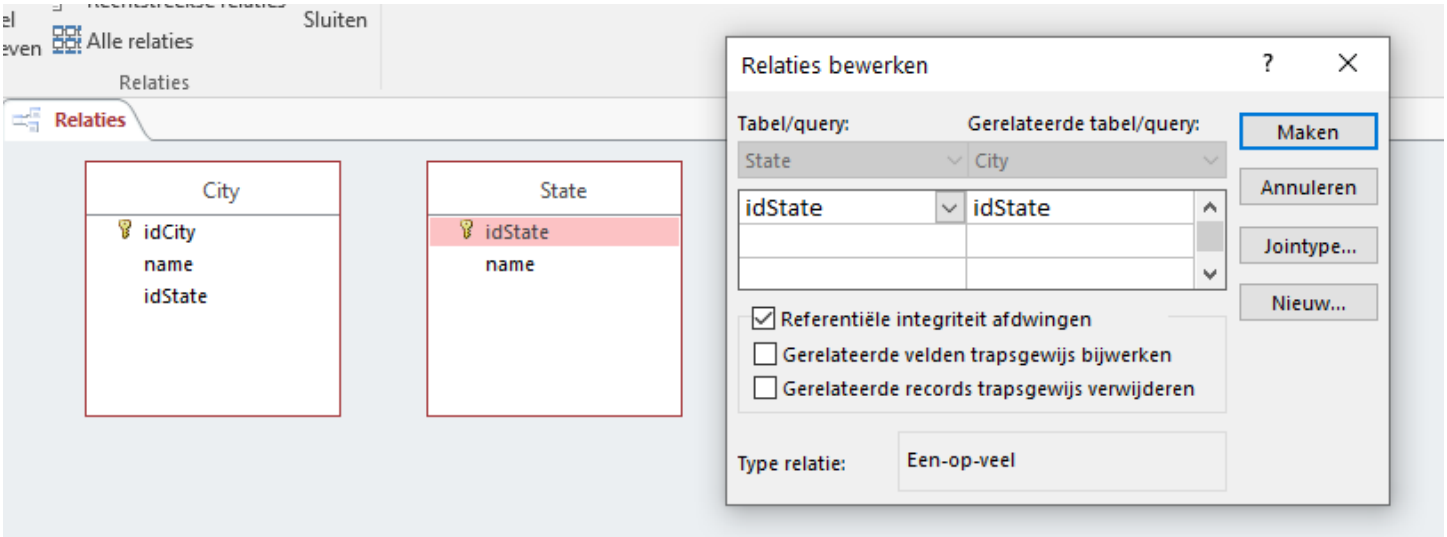
idState	name
1	Zuid-Holland
2	Noord-Holland
3	Noord-Brabant
4	Zeeland
5	Utrecht
6	Drenthe
7	Flevoland
8	Gelderland
9	Limburg
10	Groningen
11	Friesland
12	Overijssel

City

idCity	name	idState	klik om titel toe te voegen
1	Dordrecht	1	
2	Rotterdam	1	
3	Breda	3	
4	Amsterdam	2	
5	Leeuwarden	11	
6	Groningen	10	
7	Made	3	
8	Utrecht	5	
*	(Nieuw)	0	

Het is alleen nog niet af. We kunnen in de City tabel ieder getal bij idState invullen dat we maar willen. We willen dat alleen de nummers ingevoerd kunnen worden die in de State tabel staan. We doen dit door een foreign-key (Vreemde sleutel) aan te maken. Dit is een relatie tussen een veld en een primary-key. Dit kunnen we doen bij “Hulpmiddelen voor database”, waar we de relaties kunnen definiëren.

Druk op relaties en selecteer beide tabellen. Het sleuteltje bij beide tabellen geven aan dat dat veld de primary-key is (hoofd-sleutel). Nu sleep je van de State tabel het veld idState naar de idState in de City tabel. Het venster met “Relatie bewerken” komt dan tevoorschijn. Hiermee maak je een relatie tussen de velden, en met “Referentiele integriteit afdwingen” dwing je ook dat de waarden altijd correct en aanwezig moeten zijn. Nadat je op “Maken” hebt gedrukt dan is de relatie gemaakt. Nu kan je geen City meer invoeren zodat dat er een provincie bij ingevoerd wordt.



2.2 Opdrachten

2.2.1 Personen database

Op deze manier kan je ook de “Person“-tabel en de “Gender”-tabel maken zoals weergegeven in “Figuur 2: Relatie diagram”. Zet hier ook een aantal personen met adressen in en een verwijzing naar hun gender, met de correcte idCity waarden.

2.2.2 Geboorte plaats

Bovenstaande database kunnen we ook heel simpel uitbreiden door de plaats waar de persoon geboren is toe te voegen. Hoe kan je dat op een slimme manier toevoegen.

2.2.3 Metingen in een tuinderij

Ontwerp een relationele database voor de volgende situatie.

We hebben een ruimte die vol staat met plantjes. Bij iedere groep plantjes zetten we een apparaat die verschillende metingen uitvoert. We weten de x en y positie van het apparaat. We meten de temperatuur, luchtvochtigheid, bodem-vochtigheid, CO2 waarde in de lucht, en de hoeveelheid zonlicht.

Sommige metingen worden iedere minuut doorgegeven, sommige ieder uur.

In eerste instantie is de database gemaakt door iemand die niet zoveel van databases afweet. We hebben de onderstaande database gekregen.

METINGEN

Veldnaam	Gegevenstype	Beschrijving
datumMeting	Datum/tijd	datum en de tijd van de meting
positieX	Numeriek	Positie x in het grid van de ruimte
positieY	Numeriek	positie y in het grid van de ruimte
beschrijvingPlek	Korte tekst	Korte beschrijving over het meetsysteem
temperatuur	Numeriek	veldlengte is double presission
cO2	Numeriek	
h2OLucht	Numeriek	
h2OBodem	Numeriek	
licht	Korte tekst	

Maak een correcte en slimme relationele database waar deze gegevens in staan. Let hierbij op de volgende zaken. Zo min mogelijk dubbele informatie. Per apparaat wordt niet alle gegevens tegelijk meegestuurd. De ene keer krijg je temperatuur binnen, de andere keer licht. Als je per keer de gegevens in de database zet krijg je allemaal lege velden. Dat is niet wenselijk. Verzin hier wat slims voor.