Hoofdstuk 1
Databases, databasemodellen en databasesystemen

Doelen

Na het doornemen van dit hoofdstuk:

- ben je vertrouwd met een aantal basisconcepten van databases;
- weet je wat een databasemodel en een databaseschema zijn;
- heb je inzicht in de typische stappen die belangrijk zijn bij databasebeheer;
- ken je de hoofdcomponenten en basisfunctionaliteiten van een databasesysteem;
- weet je wat een databasemanagementsysteem is en hoe het typisch is opgebouwd;
- ken je de rol en het belang van databasegebruikers;
- weet je waarom databasesystemen relevant zijn.

Inleiding

Dit hoofdstuk gaat dieper in op databases, databasemodellen en databasesystemen. Deze spelen een centrale en vaak cruciale rol bij het digitaal databeheer van vrijwel alle organisaties. Inzicht in de gebruikte terminologie, de belangrijkste componenten, de basisprincipes en de basiswerking van een databasesysteem is onontbeerlijk als basis voor het begrijpen, het opzetten van, het leren omgaan met en het werken met databases. Iedereen die te maken heeft met databases en deze beter wil doorgronden – van gebruiker tot applicatieontwikkelaar, van databaseontwerper tot databaseadministrator tot ICT-manager – is daarom gebaat bij een degelijke basiskennis. Dit hoofdstuk geeft je de nodige basisinzichten en een houvast voor het begrijpen en verwerken van dit boek.

Het hoofdstuk is als volgt opgebouwd: in paragraaf 1.1 introduceren we enkele basisconcepten. Zo leren we je onder meer wat we verstaan onder data, informatie en kennis, beschrijven we het verschil tussen gestructureerde en ongestructureerde data en leggen we uit wat databases, databasemodellen en databaseschema's zijn. Daarna bespreken we in paragraaf 1.2 de belangrijkste componenten van een databasesysteem, namelijk de data, de hardware en de software. Bij data leggen we je uit wat genormaliseerde data zijn, hoe je het best omgaat met afgeleide data en wat het belang en de typische onderdelen zijn van datavoorbereiding. Bij hardware besteden we veel aandacht aan de manier waarop een databasesysteem het computergeheugen gebruikt. Bij software gaat de meeste aandacht naar het databasemanagementsysteem, meer specifiek naar de functionaliteiten en de architectuur van een databasemanagementsysteem. In paragraaf 1.3 leggen we je uit waarom gebruikers en gebruikersprofielen belangrijk zijn. Tot slot gaan we in paragraaf 1.4 dieper in op de relevantie van databasesystemen.

Elk van deze paragrafen is nodig om dit boek beter te kunnen begrijpen. Het verschil kennen tussen data, informatie en kennis helpt je om af te bakenen waarvoor een database dient en waarvoor niet. Verder is het belangrijk dat je een duidelijk onderscheid kan maken tussen een database, een databaseschema, een databasemodel, een databasemanagementsysteem en een databasesysteem. Hoewel dit boek je vooral wil aanleren hoe je met de belangrijkste softwarecomponent, namelijk een databasebeheersysteem, werkt, heb je toch enige notie nodig van hoe de andere softwarecomponenten en hardwarecomponenten werken. De basisinformatie die we hierover geven in dit hoofdstuk zul je later nodig hebben om de meer geavanceerde aspecten van databasebeheersystemen te kunnen doorgronden.

1.1 Enkele basisconcepten

Onze maatschappij is zodanig geëvolueerd dat de data en informatie waarover we beschikken, en meer nog de efficiëntie waarmee we met deze data en informatie omgaan in grote mate ons succes bepalen bij het handelen en nemen van beslissingen. Het *registreren* van informatie met het oog op permanente conservatie en verbeterde informatieoverdracht kenmerkt de ontwikkeling van de menselijke beschaving en ligt aan de basis daarvan.

Sinds de codex van Hammurabi, die dateert van circa 1780 voor Christus en die inscripties bevat over Babylonische rechtscasussen, waren steen, hout, perkament en papier achtereenvolgens de dominante informatiedragers. Deze informatiedragers konden slechts een beperkte hoeveelheid data bevatten en zijn doorgaans niet gemakkelijk aanpasbaar en doorzoekbaar. Dankzij computersystemen, meer precies *databasesystemen*, zijn we nu in staat om grotere, vaak diverse en complexe collecties van informatie digitaal te registreren, beheren en beter toegankelijk te maken.

1.1.1 Data, informatie en kennis

Om te begrijpen wat een databasesysteem is, is het belangrijk dat je het verschil kent tussen de termen data (of gegevens) en informatie.

Met **data** bedoelen we gegeven feiten. Hiermee verwijzen we naar de Latijnse betekenis van het woord *dare* (= geven) waarvan het woord data is afgeleid. Zo zijn de cijfers, getallen, symbolen, karakters en woorden die je dagelijks tegenkomt verschillende vormen van data.

VOORBEELD 1.1

Voorbeelden van data zijn de woorden *Rotterdam*, *Vissershuis*, *Monet* en het getal *1882*.

Op zich zeggen data niet alles. Je hebt ook hun betekenis en context nodig om er probleemloos mee aan de slag te kunnen. Pas dan krijg je informatie. **Informatie** verwijst dus naar data met hun betekenis en context. Informatie omvat data.

VOORBEELD 1.2

Een voorbeeld van informatie is vervat in de zin: 'In het museum Boijmans Van Beuningen in Rotterdam bevindt zich het doek *Vissershuis* dat in 1882 door Monet werd geschilderd.'

Informatie verkrijg je normaal gesproken door dingen te observeren, te lezen of je zintuigen te gebruiken. Informatie kun je ook omschrijven als het antwoord op een of meer wie-, wat-, waar- of wanneer-vragen.

Door informatie te analyseren kun je zoeken naar patronen, regels en verbanden. Dit leidt ons naar antwoorden op hoe- en waarom-vragen en noemen we **kennis**. Kennis omvat informatie, maar ook patronen, regels en verbanden waarmee je nieuwe informatie uit bestaande informatie kan afleiden.

VOORBEELD 1.3

Door informatie over de bezoeken aan het museum Boijmans Van Beuningen te analyseren kun je te weten komen welk 'type' bezoeker het meest is geïnteresseerd in het impressionisme of hoeveel tijd een gezin met kinderen gewoonlijk spendeert in het museum. Deze kennis kun je bijvoorbeeld gebruiken om toekomstige bezoekers beter te begeleiden.

Databasesystemen zijn bedoeld voor informatiebeheer. Daarbij bewaar je een aantal *data*-elementen en streef je ernaar om de betekenis en context van deze data enigszins te verduidelijken via beschrijvende data die we *metadata* noemen. Hoewel je bij dit bewaarproces probeert om *informatieverlies* zo veel mogelijk te vermijden, is dit voor complexere informatie zeer moeilijk.

1.1.2 Gestructureerde en ongestructureerde data

Wanneer de data in een databasesysteem zijn gestructureerd volgens een vastgelegd datamodel spreken we van **gestructureerde data**. Werken met gestructureerde data heeft voor- en nadelen. Met zijn werk *Phusike akroasis* (Fysica) was Aristoteles (384-322 voor Christus) een van de eersten die data gestructureerd ordende. Ordenen maakt de data overzichtelijk en efficiënter doorzoekbaar. Je moet uiteraard wel een inspanning doen om je data gestructureerd en in het juiste formaat te krijgen. Dat kan best wat tijd en voorbereiding in beslag nemen. Bovendien is veel informatie niet zonder *informatieverlies* om te zetten naar een gestructureerd formaat.

De meeste data in een computersysteem zijn echter ongestructureerd. Voorbeelden van **ongestructureerde data** zijn sensordata die afkomstig zijn van allerlei sensoren uit apparaten die aangesloten zijn via het zogenaamde *Internet of Things* (IoT) en gedigitaliseerde multimedia zoals grafieken, beelden, foto's, audio en films. Tekstuele data zoals WhatsApp-berichten, blogs, e-mails, html-bestanden en tekstdocumenten rekenen we ook tot de ongestructureerde data. Meestal kun je in tekstuele data nog een beperkte vorm van structuur herkennen, zoals titels, secties en paragrafen. In die gevallen spreken we soms ook van *semigestructureerde data*. Het vormt een uitdaging om ongestructureerde data om te zetten naar gestructureerde data zonder daarbij te veel informatie te verliezen. Een andere grote uitdaging is om ongestructureerde data te doorzoeken op basis van inhoud, in plaats van op basis van metadata. Dit noemen we *inhoud gebaseerd zoeken*. Je kunt dit tegenwoordig al onder meer toepassen op beelden, geluid en tekst.

1.1.3 Databases, databasemodellen en databaseschema's

Zoals je al las beheren we informatie in een databasesysteem door representatieve dataelementen te kiezen en deze te bewaren. Daarnaast proberen we om ook de betekenis en context van deze data te verduidelijken via metadata. Hoe dit precies gebeurt, leggen we later uit.

We kunnen nu de term *database* als volgt omschrijven:

> Een **database** is een collectie van persistente data.

Het woord persistent geeft hierbij aan dat de data gedurende een langere tijd worden bewaard in het permanent geheugen van een computersysteem. Daarmee bedoelen we geheugen waarvan de inhoud niet verloren gaat bij het uitschakelen van het systeem. Praktisch gebruiken we veelal SSD-geheugen (Solid State Drive) en soms ook HDD-geheugen (Hard Disk Drive). In normale omstandigheden blijven de data in het permanent geheugen bewaard tot ze expliciet worden gewist, door tussenkomst van een gebruiker of toepassing.

We spreken algemeen van een datacollectie omdat we niet aannemen dat de bewaarde data uniek (dus geen dataverzameling) en/of gesorteerd moeten zijn (dus geen datalijst).

Indien het niet expliciet anders is aangegeven, veronderstellen we dat de data in een database actueel zijn. Databases gebruiken we daarom meestal voor de ondersteuning van operationele taken.

VOORBEELD 1.4

Voorbeelden van databases zijn een patiëntendatabase in een huisartsenpraktijk of ziekenhuis, een productiedatabase in een bedrijf, een collectiedatabase in een museum, een geografische database in een geografisch informatiesysteem, een bibliotheekdatabase, een ledendatabase van een vereniging en een adressendatabase voor thuisgebruik.

Om te weten hoe databases in een databasesysteem worden opgebouwd en om met deze databases aan de slag te gaan, dien je het onderliggende *databasemodel* te begrijpen. Alvorens uit te leggen wat een databasemodel is, introduceren we het concept *datamodel*.

> Een **datamodel** is een geheel van voorschriften en regels om de structuur en het gedrag vast te leggen van data die in bepaalde software worden gebruikt.

In de eenvoudigste vorm bestaat het datamodel van software uit de voorschriften voor de specificatie en het gebruik van de datatypes die worden ondersteund. In de context van databasesoftware verruimen we het concept datamodel tot het concept databasemodel.

> Een **databasemodel** is het geheel van voorschriften en regels om de structuur en het gedrag van een database vast te leggen en daarnaast ook de data-integriteit te beschermen.

Met de structuur van een database bedoelen we de abstracte concepten en de datatypes waarmee we de data en de verbanden tussen data uit de database kunnen modelleren. Het *gedrag* van een database omvat de operaties waarmee we een database kunnen opzetten, de operaties waarmee we data aan een database kunnen toevoegen, verwijderen of aanpassen en de operaties om een database te doorzoeken. *Data-integriteit* slaat ten slotte op faciliteiten die ervoor zorgen dat de data zo veel mogelijk correct blijven.

Elk databasesysteem is opgebouwd naar een onderliggend databasemodel, wat wil zeggen dat de databases die worden beheerd door het databasesysteem alle moeten voldoen aan dat databasemodel. Momenteel is het *relationeel databasemodel* (Codd, 1970) het meest gebruikte. Een databasesysteem dat is gebouwd rond het relationeel databasemodel noemen we een *relationeel databasesysteem*. We behandelen het relationeel databasemodel uitgebreid in hoofdstuk 2. Andere databasemodellen komen aan bod in hoofdstukken 9 en 10.

> De meeste databasemodellen schrijven voor dat een database wordt opgebouwd volgens een vast schema. We spreken dan van een vast **databaseschema**.

Bij databasemodellering werk je rond entiteiten. Een **entiteit** kun je zien als iets waarover je informatie wil bewaren in jouw database. Voorbeelden van entiteiten zijn het schilderij *De Nachtwacht*, de kunstschilder Rembrandt en het Rijksmuseum van Amsterdam. Entiteiten worden georganiseerd per type, dat we een *entiteittype* noemen. Zo heb je bijvoorbeeld entiteittypes 'schilderij', 'artiest' en 'eigenaar'.

Databasemodellering, entiteiten en entiteittypes beschrijven we uitgebreid in de hoofdstukken 11, 12 en 13 die gaan over databaseontwerp.

Wanneer je werkt met een vast databaseschema moet je vooraf in je databaseschema een structuur vastleggen voor alle entiteittypes die je nodig hebt. Daartoe gebruik je de voorschriften en regels van het gebruikte databasemodel. Bijgevolg worden de data van alle entiteiten van eenzelfde entiteittype op dezelfde wijze gestructureerd en opgeslagen in de database.

De recentere, zogenaamde NoSQL-databasemodellen laten de eis van een vast databaseschema vallen. NoSQL-databases noemen we daarom *schemaloze databases*. De data van een entiteit moeten dan niet worden omgezet naar een vaste structuur. NoSQL-databases behandelen we in hoofdstuk 9.

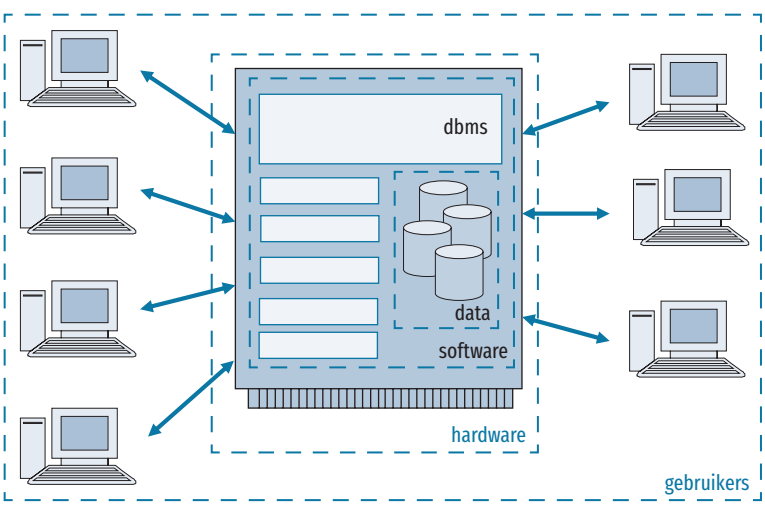
Ter afsluiting geven we graag nog mee dat vakliteratuur de term databasemodel soms ook gebruikt als alternatieve benaming voor een databaseschema. Dit leidt helaas tot verwarring, vandaar onze keuze om aparte termen te gebruiken.

1.2 Databasesystemen

We hebben in paragraaf 1.1 al een paar keer de term databasesysteem gebruikt en kunnen die nu als volgt omschrijven:

> Een **databasesysteem** is een computersysteem voor het beheer van databases.

In een *databasesysteem* onderscheiden we globaal drie *hoofdcomponenten*: de *data* (inhoud), de *software* en de *hardware*. Omdat informatie gevoelig kan zijn en vaak niet vrij mag worden gedeeld, is het belangrijk om ook de *gebruikers* van een databasesysteem te beschouwen. Figuur 1.1 stelt dit schematisch voor.



FIGUUR 1.1 *Hoofdcomponenten en gebruikers van een databasesysteem*

De drie hoofdcomponenten van een databasesysteem beschrijven we in de rest van deze paragraaf met meer diepgang. Het onderscheid tussen verschillende groepen van gebruikers behandelen we in paragraaf 1.3.

1.2.1 Data

Databases vormen de kern van een databasesysteem. Elke database sla je op in een of meer **bestanden** op permanent geheugen (SSD of HDD). Gestructureerde data worden door het databasesysteem bewaard in zogenaamde **recordbestanden**. Bij ongestructureerde data gebruikt het databasesysteem in de regel **karakterbestanden** voor tekstuele data, en **binaire bestanden** voor multimedia zoals digitale foto's en video.

Elk **record** uit een recordbestand is opgebouwd uit een of meer velden die data kunnen bevatten. Deze velden zijn vastgelegd in het **recordtype** van het record en worden elk gekarakteriseerd door een *naam* en een *datatype*. Analoog aan programmeertalen legt het datatype de toegelaten waarden en operatoren voor het veld vast. Naast velddefinities heeft een recordtype ook een naam. De veldwaarden zijn de data uit het bestand. De namen van de recordtypes en de veldnamen zijn de beschrijvende metadata: deze helpen om de betekenis en context van de data te verduidelijken. Doorgaans bevat een database verschillende records die zijn gegroepeerd volgens diverse recordtypes. Hoewel dit geen vereiste is, kunnen er in hetzelfde recordbestand records van verschillende recordtypes zijn opgeslagen.

In de praktijk kan het aantal records in een database variëren van enkele tot miljoenen of miljarden. Daarnaast kan een databasesysteem tegelijkertijd instaan voor het beheer van *meerdere* databases.

VOORBEELD 1.5

Figuur 1.2 bevat een voorstelling van de recordtypes en records uit een voorbeelddatabase voor schilderkunst. De database bestaat uit een collectie van records van drie recordtypes: *Schilderij*, *Artiest* en *Eigenaar*. Omwille van de leesbaarheid hebben we de records geordend per recordtype en laten we elke veldwaarde voorafgaan door haar corresponderende veldnaam. De gebruikte datatypes zijn:

- `CHAR(3)` voor het voorstellen van karaktersequenties (of karakterstrings) met een vaste lengte van drie karakters;
- `VARCHAR` voor het voorstellen van karaktersequenties van variabele lengte;
- `INTEGER` voor het voorstellen van gehele getallen (die geïnterpreteerd worden als jaartallen);
- `REAL` voor het voorstellen van reële getallen.

RECORDTYPE Schilderij (ID:**char(3)**; Naam:**varchar**; Artiest:**varchar**; Periode:**integer**; Waarde:**real**; Eigenaar:**varchar**)

ID:S01	Naam:Vissershuis	Artiest:Monet	Periode:1882	Waarde:16.000.000	Eigenaar:Boijmans
ID:S02	Naam:De balletklas	Artiest:Degas	Periode:1872	Waarde:8.500.000	Eigenaar:Louvre
ID:S03	Naam:Mona Lisa	Artiest:Da Vinci	Periode:1499	Waarde:75.000.000	Eigenaar:Louvre
ID:S04	Naam:Namiddag te Oostende	Artiest:Ensor	Periode:1881	Waarde:200.000	Eigenaar:KMSK

RECORDTYPE Artiest (Naam:**varchar**; Voornaam:**varchar**; Geboortejaar:**integer**; Sterfjaar:**integer**)

Naam:Da Vinci	Voornaam:Leonardo	Geboortejaar:1452	Sterfjaar:1519
Naam:Degas	Voornaam:Edgar	Geboortjaar:1834	Sterfjaar:1917
Naam:Ensor	Voornaam:James	Geboortejaar:1860	Sterfjaar:1949
Naam:Monet	Voornaam:Claude	Geboortejaar:1840	Sterfjaar:1926

RECORDTYPE Eigenaar (Naam:**varchar**; Plaats:**varchar**; Land:**varchar**)

Naam:Boijmans	Plaats:Rotterdam	Land:Nederland
Naam:Louvre	Plaats:Parijs	Land:Frankrijk
Naam:KMSK	Plaats:Antwerpen	Land:België

FIGUUR 1.2 *Records uit de voorbeelddatabase* Schilderkunst

Genormaliseerde data

Data in een database kunnen met elkaar *verbonden* zijn. Zo horen alle veldwaarden uit hetzelfde record bij elkaar. Ook tussen records kunnen er verbanden bestaan. In de database Schilderkunst is het schilderijrecord met *ID* 'S03' bijvoorbeeld gelinkt aan het artiestrecord met *Naam* 'Da Vinci'. Dergelijke verbanden tussen data helpen om de betekenis en context van de data te verduidelijken.

Databases met een vast databaseschema moeten zodanig zijn opgebouwd, dat alle verbanden die voorkomen tussen data op een efficiënte manier worden weergegeven, zonder *overtollige dataopslag* en kans op *dataverlies*. We spreken dan van **genormaliseerde data**. Data die dubbel zijn opgeslagen nemen niet alleen extra opslagruimte in beslag, maar vormen ook een potentiële bron van onjuistheden. Immers, als je bij een aanpassing niet alle kopieën aanpast, krijg je *inconsistente data*.

VOORBEELD 1.6

In de database Schilderij in figuur 1.2 wordt het verband tussen schilderijen en artiesten gemodelleerd via het veld *Artiest* van het recordtype *Schilderij* en het veld *Naam* van het recordtype *Artiest*. Een schilderijrecord met waarde *Monet* voor het veld *Artiest* wordt hierbij gerelateerd met het artiestrecord dat *Monet* als veldwaarde voor *Naam* heeft. Door data van schilderijen en artiesten met aparte recordtypes weer te geven, vermijd je overtollige dataopslag bij het modelleren van het verband tussen schilderijen en artiesten. Dit illustreert genormaliseerde data.

Wanneer je alle data van schilderijen en artiesten samenvoegt in eenzelfde recordtype, zoals geïllustreerd in figuur 1.3, krijg je overtollige dataopslag. De data over het geboorte- en sterfjaar van een artiest worden dan overtollig gekopieerd in alle schilderijrecords van die artiest. Als iemand dan het sterfjaar van Degas enkel aanpast naar 1918 in het record met *ID* 'S08', krijg je een inconsistentie met het sterfjaar 1917 in het record met *ID* 'S02'. Bovendien riskeer je zo ook ongewenst dataverlies. Immers, stel dat je alle schilderijrecords van Degas die zijn opgebouwd zoals in figuur 1.3 verwijdert uit de database, dan verlies je daarmee ook de data over het geboorte- en sterfjaar van Degas. Verwijder je alle schilderijrecords van Degas bij een opbouw zoals in figuur 1.2, dan blijven het geboorte- en sterfjaar nog beschikbaar in het artiestrecord van Degas.

RECORDTYPE Schilderij (ID:**char(3)**; Naam:**varchar**; ArtiestNaam:**varchar**; ArtiestVoornaam:**varchar**; Geboortjaar:**integer**; Sterfjaar:**integer**; Periode:**integer**; Waarde:**real**; Eigenaar:**varchar**)

ID:S02	Naam:De balletklas	ArtiestNaam:Degas	ArtiestVoornaam:Edgar	Geboortejaar:1834
Sterfjaar:1917	Periode:1872	Waarde:8.500.000	Eigenaar:Louvre	
ID:S08	Naam:Badende vrouw	ArtiestNaam:Degas	ArtiestVoornaam:Edgar	Gebootejaar:1834
Sterfjaar:1917	Periode:1886	Waarde:1.200.000	Eigenaar:Van Gogh Museum	

FIGUUR 1.3 *Een samengevoegd recordtype Schilderij*

Er bestaan ontwerptechnieken om te komen tot genormaliseerde data (Codd, 1971). Deze behandelen we uitgebreider in hoofdstuk 12. Data-integratie is en blijft een grote uitdaging. Temeer in frequent voorkomende situaties waarin je data uit verschillende databases moet samenvoegen of aan elkaar koppelen.

Geen afgeleide data

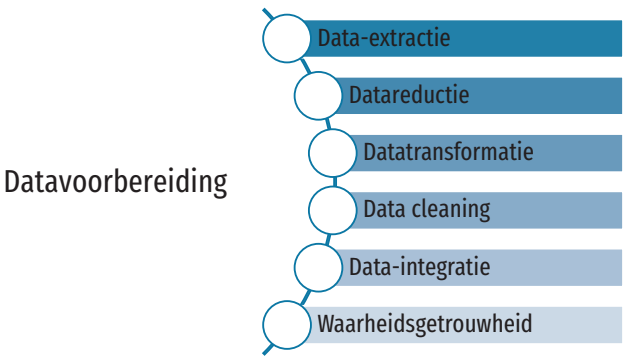
Het is belangrijk om in een database geen data op te nemen die je kunt afleiden uit andere data die in de database zitten. Doe je dit wel, dan riskeer je problemen wanneer je data aanpast en hun afgeleide data niet. Heb je bijvoorbeeld een geboortejaar en sterfjaar in je database staan en je zou ook de leeftijd bij sterfte opslaan, dan riskeer je foutieve data als je het geboorte- of sterfjaar aanpast en de leeftijd niet, of omgekeerd als je de leeftijd aanpast en het geboorte- of sterfjaar niet. Een goed databaseontwerp houdt dus rekening met normalisatie en vermijdt **afgeleide data**. Er bestaan andere technieken voor het modelleren van afgeleide data. Deze komen verderop in het boek aan bod, onder meer in paragraaf 1.2.3 bij de bespreking van de externe laag van de drielagenarchitectuur van een databasemanagementsysteem.

Datavoorbereiding

Door de grootte en complexiteit van huidige databasetoepassingen komt het vaak voor dat je data moet voorbereiden alvorens ze in een database te kunnen opnemen. Datavoorbereiding is een essentieel onderdeel van elke goede **datastrategie**, wat in essentie een langetermijnplanning is voor de technologie, processen, businessregels,

wettelijke regels en personen die vereist zijn om te kunnen voldoen aan de informatiebehoeften van een organisatie (DAMA International, 2017; Ladley, 2019; Sebastian-Coleman, 2018).

Elke informatiebehoefte is anders, ook de nodige en beschikbare databronnen variëren per situatie waarin je een database moet opzetten. Daarom is vrijwel elke datavoorbereiding anders. In figuur 1.4 geven we een overzicht van de belangrijkste aspecten die een onderdeel kunnen zijn van een **datavoorbereiding** (Reis & Housley, 2022). We beschrijven deze hier beknopt:



FIGUUR 1.4 *Mogelijke onderdelen van een datavoorbereiding*

- **Data-extractie**. Een databron, bijvoorbeeld een datastroom, bestand, tabel, tekst of website, bevat soms meer data dan nodig voor een organisatie. Het komt er dan op aan dat je een selectie maakt en de nodige data extraheert uit je databronnen om deze geëxtraheerde data in te laden in je database. Bij multimedia kan het zijn dat je eerder een aantal kenmerken wil extraheren en opslaan als metadata. Voorbeelden zijn de lengte en het genre van een lied, kenmerkende tags voor een foto of film.
- **Datareductie**. Soms volstaat data-extractie niet en moet je de geëxtraheerde data nog verder samenvatten (bij tekstuele data) of aggregeren (bij numerieke data) alvorens deze in te laden in je database. Voorbeelden van aggregatietechnieken zijn het bepalen van het minimum, het maximum of het rekenkundig gemiddelde. Zo kan het bijvoorbeeld volstaan om de gemiddelde dagtemperatuur te registreren in je database in plaats van alle temperatuurmetingen die je om het uur hebt gedaan. Dit proces van samenvatten of aggregeren noemen we *datareductie*.
- **Datatransformatie**. Data staan niet noodzakelijk in het formaat waarin je ze moet of wil opslaan in je database. Wanneer je werkt met een databaseschema is het noodzakelijk dat je data omzet naar de vaste structuren, gebruikte datatypes en integriteitsregels van dit schema. Daarnaast heb je er vaak baat bij dat je data accuraat en uniform zijn. Dit vereenvoudigt de databasemanipulaties en opzoekingen achteraf. Voor een organisatie kan het bijvoorbeeld nuttig zijn om prijsinformatie steeds met twee cijfers na de komma en in Euro voor te stellen. Het omzetten van data in de geschikte, voorgeschreven vorm noemen we *datatransformatie*.
- **Data cleaning**. Naast datatransformatie kan het ook nodig zijn om de data op te schonen en waar mogelijk te verbeteren alvorens deze in te laden in een database.