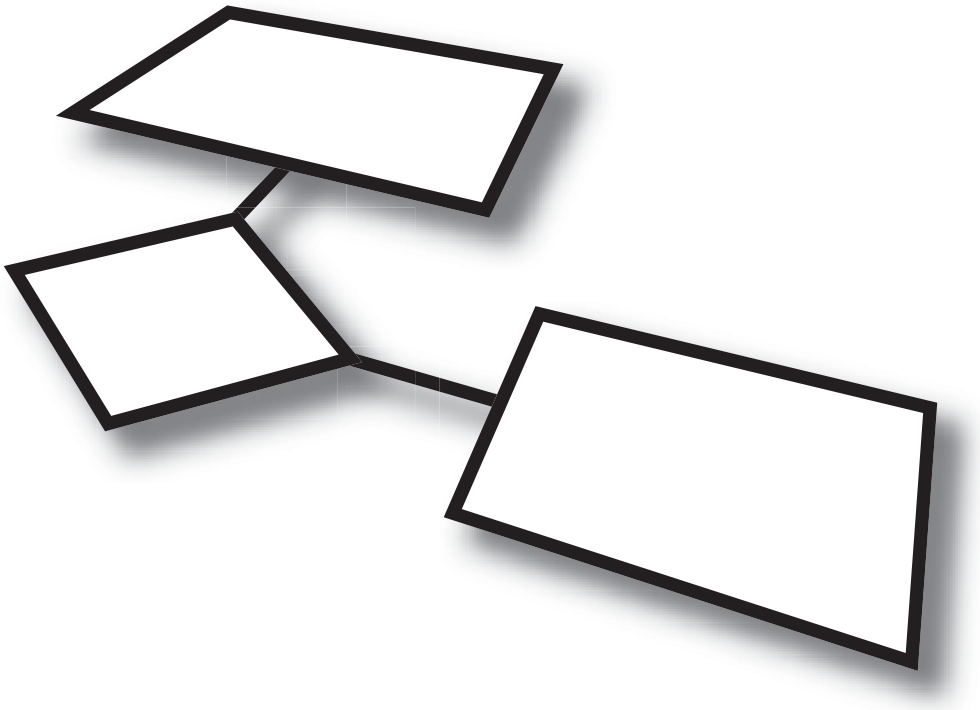
syllabus ERD
versie 1.3

Gegevensmodellering

met het entiteit-relatie-diagram



Maarten Strunk

INHOLLAND

Inhoud

1 Inleiding databases

1.1 Inleiding

Een organisatie is een samenwerkingsverband van mensen dat opgericht is voor een bepaald doel. Dit kan bijvoorbeeld een sportvereniging, een ziekenhuis, een productiebedrijf of een school zijn. Om het doel van de organisatie te bereiken, zijn naast mensen ook andere middelen nodig. Denk hierbij bijvoorbeeld aan geld, machines, gebouwen, een goede infrastructuur en kennis. Men wordt zich er steeds meer van bewust dat ook informatie een belangrijk middel is bij het nastreven van het organisatiedoel. De benodigde informatie moet op het juiste moment op de juiste plaats zijn om de organisatie goed te kunnen laten functioneren.

Doordat informatie een steeds belangrijker rol speelt, groeit de behoefte aan gegevens enorm. We kunnen op twee manieren naar de verwerking van gegevens binnen een organisatie kijken. We kunnen kijken naar de manier waarop de gegevens door de organisatie "stromen" en welke acties er mee worden uitgevoerd; de *procesgerichte* benadering, of we kunnen kijken naar de manier waarop de gegevens worden opgeslagen; de *gegevensgerichte* benadering. In deze syllabus zullen we ons met de gegevensgerichte benadering bezighouden en het ontwerpen van de manier waarop gegevens worden opgeslagen.

proces- en gegevensgerichte benadering

1.2 Databases

1.2.1 Waarom een database?

Voor de opslag van gegevens wordt er meer en meer gebruik gemaakt van de computer. Toen de automatisering nog in de kinderschoenen stond, in de jaren vijftig, werden gegevens afzonderlijk opgeslagen in aparte *bestanden*. De afdeling verkoop had aparte bestanden voor klantgegevens en artikelgegevens, de afdeling inkoop haar eigen leverancier- en artikelbestand etc. Deze manier van gegevensopslag kan tot een aantal problemen leiden. Doordat dezelfde gegevens los van elkaar worden opgeslagen op meerdere plaatsen, bestaat het gevaar dat de gegevens in het ene bestand niet meer kloppen met de gegevens in een ander bestand. Zo kan het bijvoorbeeld voorkomen dat door een inkoopfunctionaris een artikel uit het artikelbestand wordt verwijderd en deze wijziging niet wordt doorgevoerd in het artikelbestand van de verkoopafdeling. In die situatie kan het artikel dus nog steeds verkocht worden, terwijl het eigenlijk uit het assortiment is gehaald.

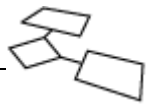
bestand

Uiteindelijk werd dit probleem opgelost door alle gegevens centraal te beheren en ze voor alle gebruikers ervan beschikbaar te maken. In plaats van bijvoorbeeld een apart artikelbestand voor elke afdeling, werd er centraal één artikelbestand bijgehouden waar alle betrokken afdelingen gebruik van konden maken. Zo'n verzameling van centraal beheerde gegevens noemen we een *database* (ook wel *gegevensbank*).

database gegevensbank

1.2.2 De structuur van een database

Bij het bouwen van een database zouden we ervoor kunnen kiezen om alle gegevens op één grote berg te gooien en op die manier te beheren. Een bestand met verkoopgegevens zou er dan uit kunnen zien zoals in figuur 1.1 is weergegeven.



VERKOPEN

Klantnaam	Klantadres	Datum	Besteld artikel	Aantal
P.Okko	Parkweg 19	12-06-2002	Sportfiets 230Z	2
P.Okko	Parkweg 19	12-06-2002	Kachel AB2	1
A.Knalbak	Akkerlaan 2	20-06-2002	Kachel AB2	4
A.Knalbak	Akkerlaan 2	20-12-2002	Kachel AB2	1
P.Okko	Parkweg 19	01-12-2002	Sportfiets 230Z	36
…	…	…	…	…

► **Figuur 1.1** Een voorbeeld van een bestand met verkoopgegevens

redundantie

Wat direct opvalt, is dat dezelfde gegevens in dit verkoopbestand meer dan eens voorkomen. Zo komt het artikel SPORTFIETS 230Z in dit voorbeeld twee keer voor, en het artikel KACHEL AB2 drie keer. Ook de klantnamen en klantadressen worden telkens herhaald. Het vaker voorkomen van dezelfde gegevens noemen we *redundantie*. Redundantie brengt een aantal problemen met zich mee.

Het meerdere keren opslaan van dezelfde gegevens kost natuurlijk veel (schijf)ruimte. Daarnaast komt het de betrouwbaarheid van de gegevens niet ten goede. Doordat de verkoopmedewerker bij het invoeren van elke verkooporder weer opnieuw de naam en het adres van de klant moet invoeren, is de kans groot dat er een keer een fout wordt gemaakt en er bijvoorbeeld P.OKKA wordt getypt. Hierdoor wordt er per ongeluk een artikel verkocht aan een klant die helemaal niet bestaat. Bovendien is het bestand op deze manier lastig te onderhouden. Stel nu bijvoorbeeld dat klant P.OKKO verhuist van PARKWEG 19 naar PERIOKOPLEIN 187B. Op elke plek waar deze klant in het verkoopbestand voorkomt (misschien wel een paar duizend keer), zal nu het klantadres moeten worden gewijzigd, waarbij de kans op fouten natuurlijk groot is.

relationele model

Om redundantie te voorkomen, zullen we voordat we beginnen met bouwen goed moeten nadenken hoe we de database gaan inrichten. In de loop der tijd zijn er verschillende oplossingen bedacht om redundantie zoveel mogelijk te beperken. In deze syllabus zullen we gebruik maken van het *relationele model* dat een oplossing biedt voor redundantie. In het relationele model worden de kolommen met gegevens die zich telkens herhalen op zichzelf staande bestanden. Figuur 1.2 toont wat dit zou betekenen voor ons voorbeeld.

KLANTEN

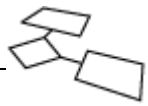
Klantnaam	*Klantadres*
P.Okko	*Parkweg 19*
P.Okko	*Parkweg 19*
A.Knalbak	*Akkerlaan 2*
A.Knalbak	*Akkerlaan 2*
P.Okko	*Parkweg 19*
…	…

ARTIKELEN

Datum	*Besteld artikel*	Aantal
12-06-2002	*Sportfiets 230Z*	2
12-06-2002	*Kachel AB2*	1
20-06-2002	*Kachel AB2*	4
20-12-2002	*Kachel AB2*	1
01-12-2002	*Sportfiets 230Z*	36
…	…	…

► **Figuur 1.2** Het verkoopbestand opdelen in een verkoop-, artikel- en klantenbestand

Er ontstaan, naast het al aanwezige verkoopbestand, twee nieuwe bestanden: het bestand KLANTEN en het bestand ARTIKELEN. In deze nieuwe bestanden komt elke klant en elk artikel slechts één keer voor. Het enige dat we nu nog maar hoeven te doen, is vanuit het bestand VERKOPEN verwijzen naar de juiste klant en het juiste artikel. Dit wordt meestal gedaan door middel van een nummer of code zoals figuur 1.3 laat zien.



KLANTEN

Klantnr.	Klantnaam	Klantadres
265	P.Okko	Parkweg 19
791	A.Knalbak	Akkerlaan 2
…	…	…

ARTIKELEN

Artikelcode	Omschrijving
SF23	Sportfiets 230Z
KAB2	Kachel AB2
…	…

VERKOPEN

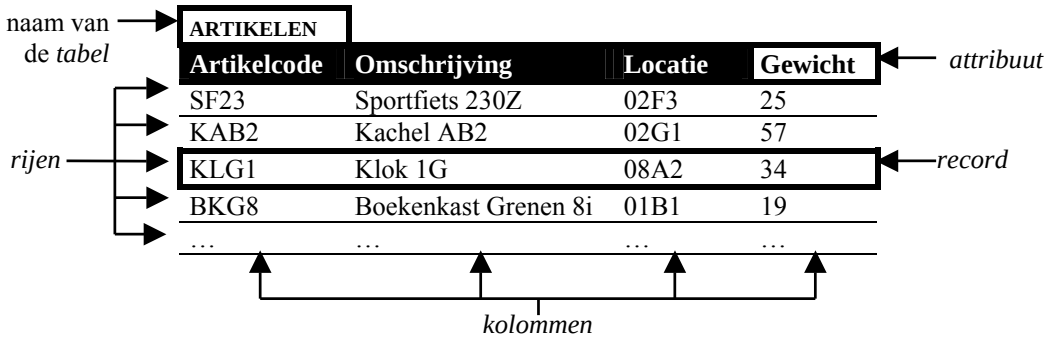
Klantnr.	Artikelcode	Datum	Aantal
265	SF23	12-06-2002	2
265	KAB2	12-06-2002	1
791	KAB2	20-06-2002	4
791	KAB2	20-12-2002	1
265	SF23	01-12-2002	36
…	…	…	…

► **Figuur 1.3** Vanuit het verkoopbestand verwijzen naar het klanten- en artikelbestand door middel van een Klantnr en een Artikelcode

1.2.3 Begrippen bij het werken met databases

tabel
attribuut
record

Tot nu toe spraken we van bestanden en kolommen. Bij het werken met databases worden echter vaak begrippen als *tabel*, *attribuut* en *record* gebruikt. Figuur 1.4 toont wat er met deze begrippen wordt bedoeld.



► **Figuur 1.4** Enkele belangrijke begrippen bij het werken met databases

Voor bovenstaande begrippen worden in de literatuur vaak synoniemen gebruikt. Hieronder is een overzicht gegeven van deze synoniemen en de betekenis die daar in deze syllabus aan wordt gegeven:

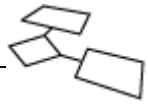
begrip	***synoniem***
tabel	bestand, entiteit, entiteittype
record	rij, regel, tupel
attribuut	kolomtype, veld, veldtype, attribuuttype

1.2.4 Sleutels

Als er, zoals in paragraaf 1.2.2 is besproken, vanuit een bepaalde tabel verwezen wordt naar een andere tabel, mag er geen misverstand bestaan over welk record er in die andere tabel bedoeld wordt. Als we vanuit de tabel VERKOPEN bijvoorbeeld verwijzen naar een klant met klantnummer 265, willen we zeker weten dat dit nummer hoort bij klant P.OKKO en dat er geen andere klanten zijn met dit nummer. Binnen de tabel KLANTEN moet elk klantnummer dus uniek zijn. Zo'n attribuut dat een record uniek maakt, noemen we een *sleutel* (ook wel: *primaire sleutel*, of in het Engels: *(primary) key*, of: *unique identifier*).

sleutel

Een sleutel kan uit één of meer attributen bestaan. Voor de tabel KLANTEN is alleen het klantnummer genoeg om ervoor te zorgen dat elk record uniek is. Maar welk attribuut zou bij de tabel VERKOPEN als sleutel kunnen dienen?



samengestelde sleutel

Geen enkel attribuut in deze tabel maakt een record uniek. Een oplossing zou kunnen zijn, om de attributen KLANTNR, ARTIKELCODE en DATUM samen de sleutel te laten vormen. De *samengestelde sleutel* die zo ontstaat, is uniek voor elk record (als we er vanuit gaan dat een klant nooit meer dan één keer op dezelfde datum hetzelfde artikel kan kopen).

In plaats van een samengestelde sleutel, hadden we er in dit voorbeeld ook voor kunnen kiezen om als sleutel een nieuw attribuut VERKOOPNUMMER toe te voegen aan de tabel VERKOPEN. Welke oplossing kunnen we nu het beste kiezen? Om redenen van efficiency wordt er in de praktijk meestal naar gestreefd de sleutel uit zo min mogelijk attributen te laten bestaan. In die zin zou de oplossing met het VERKOOPNUMMER dus het best zijn (de sleutel bestaat dan immers uit maar één attribuut in plaats van drie).

1.2.5 Referentiële integriteit

referentiële integriteit

inconsistentie

Wanneer we in de tabel VERKOPEN uit het voorbeeld een klantnummer 3006 zouden opnemen terwijl er in de tabel KLANTEN geen klant voorkomt met dat nummer, ontstaat er een probleem. In zulke gevallen zijn de verwijzingen (referenties) tussen de tabellen, en daarmee ook de gegevens, niet meer betrouwbaar (integer). Het betrouwbaar zijn van de verwijzingen in een database noemen we *referentiële integriteit*. De meeste moderne databasesoftware controleert automatisch de referentiële integriteit. Als er zich situaties voordoen waarin gegevens niet met elkaar in overeenstemming zijn, dan spreken we van *inconsistente gegevens*. Samenvattend kan dus worden gesteld dat redundantie of het ontbreken van referentiële integriteit kunnen leiden tot inconsistente gegevens.

Samenvatting

Database

In een organisatie werken mensen samen om een bepaald doel te bereiken. Om dit doel te bereiken maakt men gebruik van allerlei gegevens, die ook ergens opgeslagen moeten worden. Dit gebeurt tegenwoordig meestal in één centrale opslag: een **database**.

Databasestructuur

Wanneer alle gegevens in een database worden bewaard op één grote berg, ontstaat er al gauw **redundantie**: dezelfde gegevens komen meerdere keren voor, met alle gevaren van dien. Het **relationele model** biedt een oplossing voor dit probleem: de grote berg met gegevens wordt gesplitst in een aantal kleine bergjes: **tabellen**. Deze tabellen worden door middel van **sleutels** (nummers of codes die een **record** uniek maken) met elkaar verbonden. De verwijzingen tussen de tabellen die zo ontstaan moeten natuurlijk wel kloppen, dit noemen we **referentiële integriteit**.

Begrippen

2 Het entiteit-relatie-diagram

2.1 Inleiding

In het vorige hoofdstuk hebben we gezien welke problemen zich kunnen voordoen bij het bouwen van een database. Daarom is het verstandig om, voordat we een database bouwen, eerst een ontwerp te maken. In dit ontwerp kunnen we dan vastleggen welke tabellen er nodig zullen zijn, welke attributen deze tabellen moeten hebben en wat de structuur van de database moet worden.

Nu zouden we dit ontwerp kunnen maken in de vorm van een stuk tekst en op die manier de database beschrijven, maar dit maakt het er in de praktijk niet duidelijker op. We zouden de database liever tekenen, om zo een overzichtelijk plaatje te krijgen. Een plaatje zegt vaak meer dan duizend woorden. Voor het ontwerpen van een *relationele database* (een database die is gebaseerd op het relationele model) zijn in de loop der tijd meerdere technieken ontwikkeld. De techniek waar we in deze syllabus mee gaan werken is het *entiteit-relatie-diagram* (of afgekort: *ERD*).

relationele database

ERD

2.2 Symbolen van het ERD

Bij het tekenen van een ERD kan gebruik worden gemaakt van drie symbolen: de *entiteit*, de *relatie* en het *attribuut*. Daarnaast kan nog een aantal eigenschappen van relaties worden aangegeven: de *cardinaliteit*.

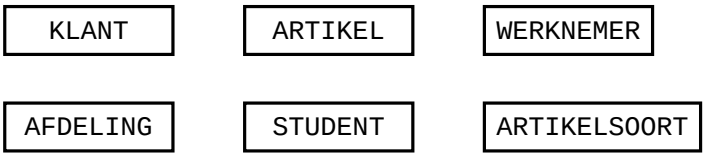
2.2.1 Entiteiten

Eén van de symbolen die we bij het tekenen van een ERD kunnen gebruiken, is de *entiteit*. Een entiteit is iets waarover we (nu of in de toekomst) gegevens willen vastleggen en dat van betekenis is voor de organisatie waar we de database voor ontwerpen. In het ERD gebruiken we een rechthoek om een entiteit aan te duiden.

entiteit

► **Figuur 2.1** Het symbool voor een entiteit

In de rechthoek zetten we de naam van de entiteit. Het is gebruikelijk om deze naam in enkelvoud te noteren. Figuur 2.2 laat een aantal voorbeelden van entiteiten zien.



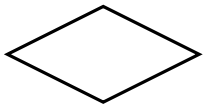
► **Figuur 2.2** Enkele voorbeelden van entiteiten

De entiteiten uit het ERD worden tabellen in de database die we uiteindelijk gaan bouwen. Een database gebouwd aan de hand van bovenstaande entiteiten zou dus een tabel KLANT, een tabel ARTIKEL, een tabel WERKNEMER, een tabel AFDELING, etc. bevatten.

2.2.2 Relaties

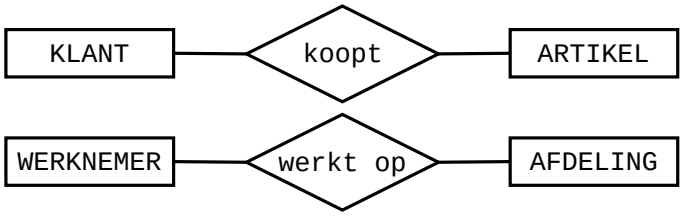
In hoofdstuk 1 hebben we gezien dat er bij een relationele database verwijzingen zijn tussen de tabellen. Zo zagen we dat een tabel met verkoopgegevens bijvoorbeeld een attribuut klantnummer bevat, dat verwijst naar een record met klantgegevens in de tabel met klanten. Deze verwijzingen kunnen ook in het ERD worden opgenomen en worden *relaties* genoemd. Een relatie is een zinvolle samenhang tussen entiteiten en wordt aangeduid met een ruit.

relatie



► **Figuur 2.3** Het symbool voor een relatie

In de ruit zetten we de naam van de relatie. Relaties verbinden entiteiten met elkaar, zoals figuur 2.4 hieronder laat zien.



► **Figuur 2.4** Twee voorbeelden van een relatie

De richting waarin we de relatie tekenen (van links naar rechts, van boven naar beneden) mogen we zelf bepalen. Zoals bovenstaand figuur laat zien wordt de naam van de relatie zo gekozen, dat van links naar rechts (of van boven naar beneden) een leesbare "zin" ontstaat. In het voorbeeld kunnen we lezen: "*een klant koopt een artikel*" en "*een werknemer werkt op een afdeling*."

In plaats van bovenstaande relatie tussen KLANT en ARTIKEL, hadden we ook het volgende kunnen tekenen:

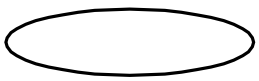
Dit is precies hetzelfde als de klant-artikel relatie in figuur 2.4. Het feit *dat* er een relatie is tussen beide entiteiten is van belang, in welke richting we die tekenen en welke naam we die precies geven, kunnen we verder zelf bepalen.

In de uiteindelijk te bouwen database worden sommige relaties tabellen, andere relaties verdwijnen bij de vertaling naar relationeel model. In hoofdstuk 3 zullen we hier dieper op ingaan.

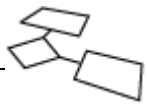
2.2.3 Attributen

attribuut

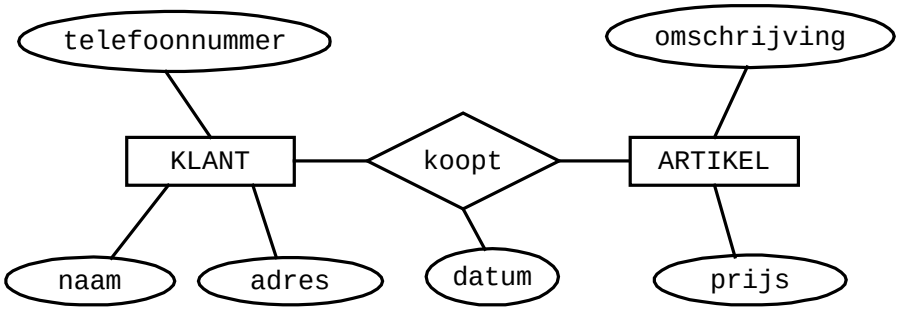
Attributen zijn eigenschappen die we van een entiteit (of relatie) willen bijhouden. Als een organisatie bijvoorbeeld geïnteresseerd is in de naam, het adres en het telefoonnummer van hun klanten en dit wil bijhouden in de database, hebben we in het ontwerp een entiteit KLANT nodig met daaraan de attributen NAAM, ADRES en TELEFOONNUMMER. Een attribuut wordt in het ERD aangeduid met een ellips.



► **Figuur 2.5** Het symbool voor een attribuut



In de ellips zetten we de naam van het attribuut. Zowel entiteiten als relaties kunnen attributen bevatten, zoals onderstaand voorbeeld laat zien.



► **Figuur 2.6** Enkele voorbeelden van attributen

Niet alle relaties hoeven attributen te hebben. Entiteiten hebben wel altijd één of meer attributen. Als een entiteit geen attributen heeft willen we er blijkbaar niets van bijhouden en is het dus meestal zinloos om deze "lege" entiteit in het ontwerp op te nemen.

attributenlijst

Door attributen als ellips in het ERD op te nemen, wordt het ontwerp al gauw onoverzichtelijk omdat ze veel ruimte innemen, zeker als het ontwerp ingewikkeld is en er veel attributen zijn. Daarom is het gebruikelijk om de attributen niet in het schema weer te geven, maar in een apart overzicht bij het ERD te leveren. Figuur 2.7 laat een voorbeeld zien van zo'n *attributenlijst*. In deze syllabus zullen beide notaties door elkaar worden gebruikt.

Attributenlijst

```
KLANT       : naam, adres, telefoonnummer
ARTIKEL     : omschrijving, prijs
koopt       : datum
```

► **Figuur 2.7** Voorbeeld van een attributenlijst

2.3 Cardinaliteit

cardinaliteit

Om een goede database te kunnen bouwen, willen we in de praktijk meestal meer weten van een relatie dan tot nu toe is beschreven. Deze extra informatie die we van een relatie vastleggen, noemen we *cardinaliteit* en kan worden onderverdeeld in *functionaliteit* en *totaliteit*.

2.3.1 Functionaliteit

Bekijk onderstaand voorbeeld.

► **Figuur 2.8** Een ontwerp voor een database waarin docenten en cursussen worden bijgehouden, en door de relatie tevens welke docent welke cursus geeft

De relatie tussen DOCENT en CURSUS in dit voorbeeld kan op meerdere manieren worden geïnterpreteerd:

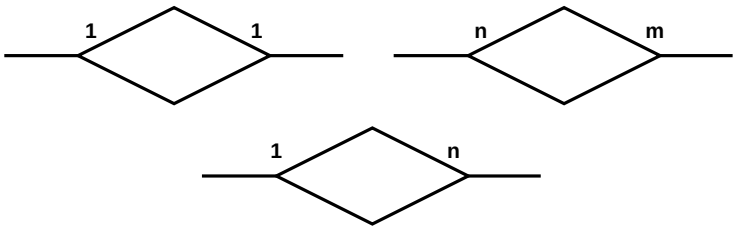
- Een bepaalde docent kan meerdere cursussen geven
- Een bepaalde docent kan niet meer dan één cursus geven
- Een bepaalde cursus kan door meerdere docenten gegeven worden
- Een bepaalde cursus kan door niet meer dan één docent gegeven worden

Het is belangrijk om in ons ontwerp vast te leggen van welke interpretatie er in de organisatie waar we de database voor ontwerpen sprake is. Stel dat we bovenstaand ERD hebben ontworpen voor een school waar alleen gespecialiseerde docenten werken, die maar één cursus kunnen geven, maar waar een bepaalde cursus wel door meerdere van die docenten kan worden gegeven. Deze situatie kan van grote invloed zijn op de structuur van de database en kunnen we in een ERD als volgt vastleggen:

► **Figuur 2.9** Een ERD waarin de functionaliteit is aangegeven

functionaliteit

Het maximum aantal keer dat de ene entiteit kan voorkomen in relatie tot de andere, noemen we de *functionaliteit*. Wanneer een bepaalde entiteit maximaal één keer kan voorkomen in relatie tot een andere entiteit, noteren we een `1` bij de relatie, aan de kant van die entiteit. Als zij meerdere keren kan voorkomen, noteren we een `n`. Op die manier ontstaan de volgende mogelijke combinaties:



► **Figuur 2.10** De verschillende soorten functionaliteit

De functionaliteit van een relatie wordt bepaald door de manier waarop de organisatie waarvoor we een ontwerp maken werkt, en moet dus voor elke organisatie opnieuw bekeken worden. Op de ene school wordt bijvoorbeeld een cursus door meerdere docenten gegeven (`n`), op een andere altijd door maximaal één (`1`). Dit resulteert dus in twee verschillende ontwerpen.

Om verwarring bij het bepalen van de functionaliteit te voorkomen, kunnen het best de volgende zinnen worden gebruikt:

(`1`) "*Een bepaalde ... kan één keer voorkomen in relatie tot een ...*"
(`n`) "*Een bepaalde ... kan meerdere keren voorkomen in relatie tot een ...*"

waarbij op de puntjes (...) de namen van de betreffende entiteiten worden ingevuld. Hieronder twee voorbeelden.

Lees:

- Een bepaalde DOCENT kan meerdere CURSUSsen geven
- Een bepaalde CURSUS kan door maximaal één DOCENT worden gegeven

Lees:

- Een bepaalde WERKNEMER kan op meerdere AFDELINGen werken
- Op een bepaalde AFDELING kunnen meerdere WERKNEMERS werken

► **Figuur 2.11** Twee voorbeelden van hoe functionaliteit kan worden gelezen

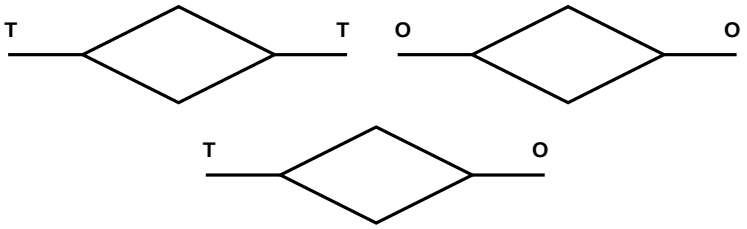
2.3.2 Totaliteit

Een andere vraag die we ons kunnen stellen bij de relatie uit figuur 2.8, is of *iedere* docent een cursus geeft, of dat er ook docenten zijn die geen cursussen geven (maar bijvoorbeeld alleen onderzoek doen). We kunnen de volgende interpretaties geven aan de relatie uit figuur 2.8:

- Iedere docent geeft cursussen
- Niet iedere docent geeft cursussen
- Iedere cursus wordt gegeven door een docent
- Niet iedere cursus wordt gegeven door een docent

totaliteit

Het minimum aantal keer dat de ene entiteit kan voorkomen in relatie tot de andere, noemen we de *totaliteit*. Wanneer een bepaalde entiteit altijd minimaal één keer voorkomt in relatie tot een andere entiteit, noteren we een T (van "totaal") aan de kant van die entiteit. Als zij niet altijd voorkomt, noteren we een O (van "optioneel"). Op die manier ontstaan de volgende mogelijk combinaties:



► **Figuur 2.12** De verschillende soorten totaliteit

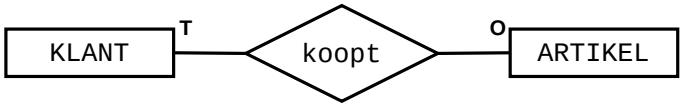
Ook bij het bepalen van de totaliteit kunnen we gebruik maken van hulpzinnen:

(T) "*Voor iedere ... geldt dat deze voorkomt in relatie tot ...*"
(O) "*Niet iedere ... komt voor in relatie tot ...*"

waarbij op de puntjes (...) de namen van de betreffende entiteiten worden ingevuld. Hieronder twee voorbeelden.

Lees:

- Iedere WERKNEMER werkt op een AFDELING
- Op iedere AFDELING werken WERKNEMERS



Lees:

- Iedere KLANT heeft een ARTIKEL gekocht
- Niet ieder ARTIKEL is verkocht aan een KLANT

► **Figuur 2.13** Twee voorbeelden van hoe totaliteit kan worden gelezen

Voor de totaliteit geldt hetzelfde als voor de functionaliteit: zij wordt bepaald door de situatie in de organisatie waar we ons ontwerp voor maken. In de ene organisatie zal iedere klant een artikel hebben gekocht, bijvoorbeeld omdat klanten pas worden geregistreerd op het moment dat ze iets kopen (T). Een andere organisatie houdt een groot klantenbestand bij met daarin ook alvast klanten die (nog) niets gekocht hebben (O).

2.4 Historische en actuele databases

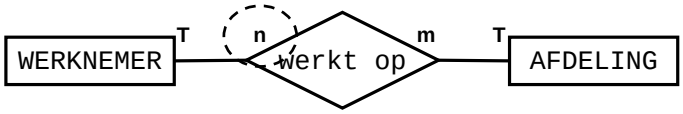
historische database

actuele database

Er zijn databases die zowel de gegevens van dit moment bijhouden, als een historisch overzicht. Zulke databases noemen we *historische databases*. Dit type database komt veruit het meest voor, omdat organisaties vaak geïnteresseerd zijn in gegevens uit het verleden. Denk bijvoorbeeld aan de verkoopresultaten van het afgelopen jaar in een handelsbedrijf. Om zulke gegevens uit de database te kunnen halen, moeten deze er in de loop der tijd wel in zijn bewaard. Daar tegenover staat de *actuele database* waarin alleen de gegevens van dit moment worden vastgelegd.

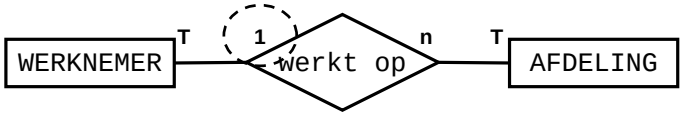
In deze syllabus zullen we meestal uitgaan van een historische database, omdat dit in de praktijk de meest voorkomende situatie is. Het feit of de database actueel of historisch moet worden, kan effect hebben op de cardinaliteit van het ERD dat we ervoor maken. Onderstaand voorbeeld illustreert dit.

> Bij het bedrijf *Crocon b.v.* werken werknemers altijd op één afdeling tegelijk. Op een afdeling werken altijd meerdere werknemers.



Functionaliteit bij een historische database, lees:

- Een bepaalde WERKNEMER kan *in de loop der tijd* op meerdere AFDELINGen hebben gewerkt



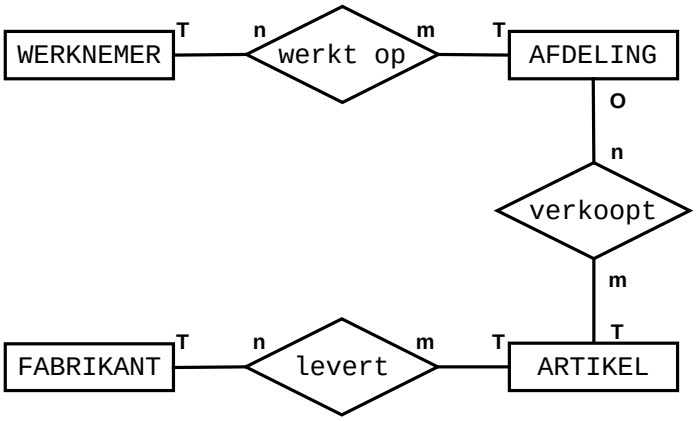
Functionaliteit bij een actuele database, lees:

- Een bepaalde WERKNEMER werkt *momenteel* op één AFDELING

► **Figuur 2.14** Het verschil in functionaliteit tussen een historische en actuele database

2.5 Voorbeeld van een ERD

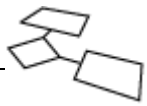
Om de tot nu toe behandelde theorie te demonstreren volgt hieronder een voorbeeld van een ERD.



Attributenlijst

```
WERKNEMER   : naam, voornaam, salaris
AFDELING    : naam, telefoonnummer
ARTIKEL     : omschrijving, verkoopprijs
FABRIKANT   : naam, adres, plaatsnaam,
              postcode, telefoonnummer
werkt op    : begindatum
levert      : datum, inkoopprijs
```

► **Figuur 2.15** Voorbeeld van een ERD



Figuur 2.15 is het ontwerp van een (historische) database voor een organisatie die als volgt werkt (ga dit zelf na):

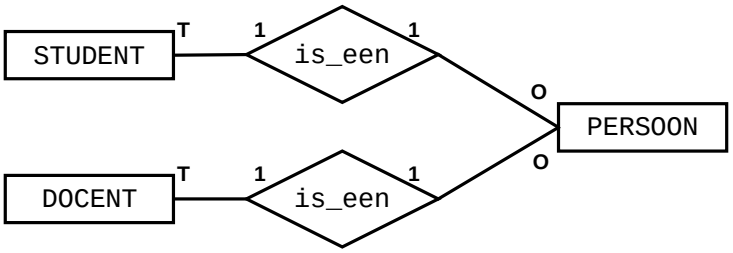
> Werknemers kunnen in de loop der tijd op meerdere afdelingen werken en werken altijd op een bepaalde afdeling. Op een afdeling werken altijd meerdere weknemers. Een aantal afdelingen verkoopt artikelen. Alle artikelen kunnen worden geleverd door meerdere fabrikanten en worden verkocht door meerdere afdelingen. Een fabrikant kan meerdere soorten artikelen leveren en wordt pas in het fabrikantenbestand opgenomen wanneer er voor het eerst artikelen bij hem worden besteld.
>
> Van werknemers wil men graag de voornaam, naam en het salaris bijhouden en vanaf welke datum ze op een bepaalde afdeling hebben gewerkt. Het is in de loop der tijd handig gebleken, om per afdeling een naam en (intern) telefoonnummer bij te houden. Van artikelen houdt men altijd de omschrijving en de verkoopprijs bij, van fabrikanten de naam van het bedrijf, het adres (inclusief plaatsnaam en postcode) en het telefoonnummer. Tot slot houdt men altijd bij op welke datum welke artikel is ingekocht bij welke leverancier en tegen welke inkoopprijs dit is gebeurd.

2.6 Bijzondere relaties

2.6.1 Generalisatie

Het komt wel eens voor dat twee (of meer) entiteiten in een ERD voor een groot deel exact dezelfde attributen hebben. Denk hierbij bijvoorbeeld aan de entiteiten DOCENT en STUDENT. Beide entiteiten kunnen de attributen naam, voornaam, adres, woonplaats, postcode, telefoonnummer, e-mail en geboortedatum hebben (tevens kan van een DOCENT nog het salaris worden bijgehouden en van een STUDENT het studentnummer). Met een ERD proberen we een zo efficiënt mogelijke database te ontwerpen, en twee entiteiten met zoveel dezelfde attributen is niet erg efficiënt te noemen. De oplossing hiervoor wordt *generalisatie* genoemd en ziet er uit zoals in figuur 2.16 wordt weergegeven.

generalisatie



Attributenlijst

```
PERSOON     : naam, voornaam, adres,
              woonplaats, postcode,
              telefoonnummer, e-mail,
              geboortedatum
STUDENT     : studentnummer
DOCENT      : salaris
```

► **Figuur 2.16** Een voorbeeld van generalisatie

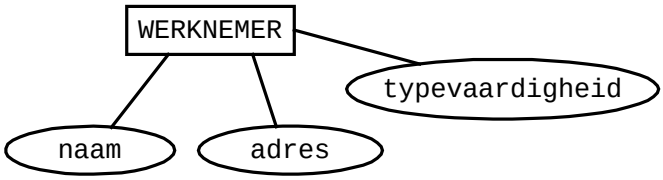
Alle attributen die in zowel STUDENT als DOCENT voorkomen, worden samengevoegd in een nieuwe entiteit PERSOON, zodat deze attributen maar één keer in het ERD voorkomen. In STUDENT en DOCENT blijven alleen de attributen staan die specifiek voor die entiteit gelden (en dus niet zijn samen te voegen in PERSOON). In het voorbeeld zijn dat SALARIS en STUDENTNUMMER.

De functionaliteit en totaliteit zijn bij generalisatie *altijd* hetzelfde, namelijk een 1-op-1 relatie, met een O aan de kant van de generalisatie en een T aan de andere zijde van de relatie. De naam van de relatie, *IS_EEN*, is een gereserveerde naam en mag alleen gebruikt worden bij een bijzondere relatie (zoals hier bij generalisatie). Een IS_EEN relatie heeft nooit attributen.

is_een relatie

2.6.2 Specialisatie

Bekijk onderstaand voorbeeld.



► **Figuur 2.17** Entiteit werknemer met de attributen naam, adres en typevaardigheid

De database die op basis van dit ontwerp gebouwd zal worden, zal een tabel WERKNEMER bevatten met in elk geval de attributen NAAM, ADRES en TYPEVAARDIGHEID. Stel dat de organisatie waar dit ontwerp voor is gemaakt 1500 werknemers heeft, waarvan er 25 typist zijn. In de tabel WERKNEMER zullen dan 1500 records worden opgenomen, waarbij van maar liefst 1475 records het attribuut TYPEVAARDIGHEID leeg zal worden gelaten.

WERKNEMER

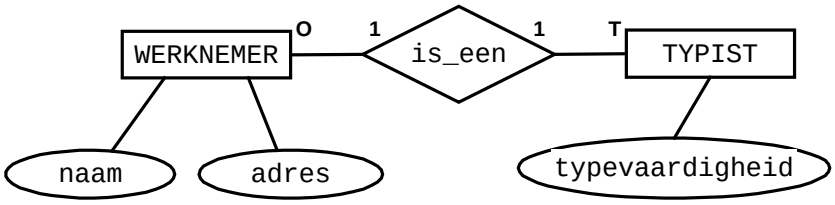
Naam	Adres	Typevaardigheid
Arie	Arielaan 6	
Piet	Pietaria 89b	
Jan	Jweg 17	
Klaas	Klastraat 2	
Miep	Mistraat 90	Typediploma B
Jannie	Janneplein 8	
…	…	…

Van de 1500 records zullen er 1475 van deze lege velden bevatten

► **Figuur 2.18** Voorbeelden van enkele records in de tabel werknemer

specialisatie

Een dergelijke tabel neemt natuurlijk veel (schijf)ruimte in beslag en is bovendien niet erg efficiënt in gebruik. Om dit probleem op te lossen kunnen we in ons ERD gebruik maken van *specialisatie*. Figuur 2.19 hieronder geeft aan hoe we specialisatie in ons voorbeeld van de werknemers en typisten zouden toepassen.



► **Figuur 2.19** Een voorbeeld van specialisatie

We voegen een nieuwe entiteit TYPIST toe, die een specialisatie is van de entiteit WERKNEMER. Alle attributen die alleen op typisten van toepassing zijn, nemen we vervolgens op in TYPIST. Op deze manier wordt de tabel WERKNEMER uit figuur 2.18 dus gesplitst in twee tabellen: één met de gegevens van alle werknemers, en één met de typistgegevens van een aantal bijzondere werknemers: de typisten. De tabel WERKNEMER zal dan 1500 records bevatten, en de tabel TYPIST 25 records. In beide tabellen zullen geen lege velden meer voorkomen.

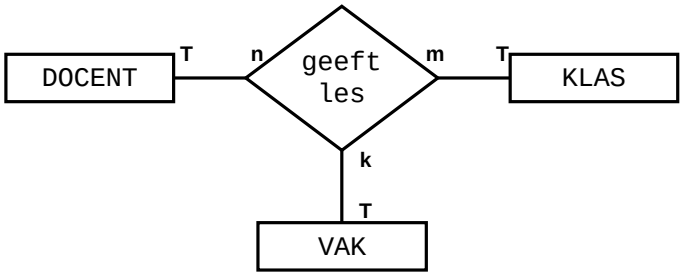
Bij specialisatie wordt, net als bij generalisatie, gebruik gemaakt van een IS_EEN relatie. Ook hier heeft die relatie *altijd* dezelfde functionaliteit (1-op-1) en totaliteit (T aan de kant van de specialisatie, O aan de andere kant van de relatie).

2.6.3 Meerdimensionale relaties

Tot nu toe zijn we ervan uitgegaan dat een relatie altijd twee entiteiten verbindt. Soms is het echter handig om een relatie drie of zelfs meer entiteiten met elkaar te laten verbinden. Bekijk het volgende voorbeeld.

► **Figuur 2.20** Iedere docent geeft les aan één of meer klassen, iedere klas krijgt les van één of meer docenten

Bovenstaand ERD resulteert in een database waarin we gegevens kunnen vastleggen over docenten en klassen, en dankzij de relatie tevens kunnen bijhouden welke docent lesgeeft aan welke klas. Stel nu dat we bovendien nog willen bijhouden welke docent aan welke klas welk *vak* geeft. De relatie zal dan niet alleen de entiteiten DOCENT en KLAS met elkaar moeten verbinden, maar ook de (nieuwe) entiteit VAK. Figuur 2.21 laat zien hoe dit gaat.



► **Figuur 2.21** Een voorbeeld van een driedimensionale relatie

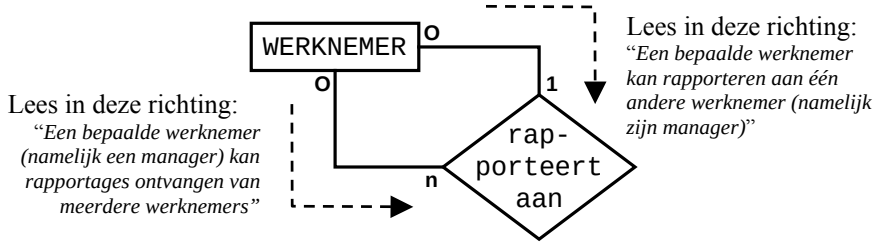
meerdimensionale relatie

De relatie uit figuur 2.21 noemen we een *meerdimensionale relatie*, omdat zij meer dan twee entiteiten verbindt. In dit geval is er eigenlijk sprake van een *driedimensionale relatie*. Op dezelfde wijze kunnen we ook vier, vijf of nog meer entiteiten met elkaar verbinden. Bij het kiezen van de naam van zo'n relatie, moeten we aanhouden dat deze zo duidelijk mogelijk de samenhang tussen de entiteiten weergeeft (het hoeft niet per se een lopende zin te zijn).

In hoofdstuk 4 komen we nog uitgebreid terug op de manier waarop meerdimensionale relaties in de praktijk het best gebruikt kunnen worden, en welke gevolgen dit heeft voor de te bouwen database.

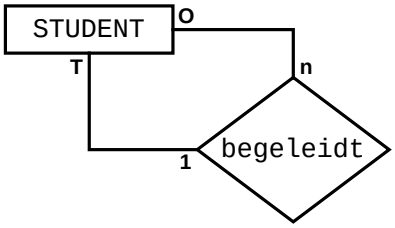
2.6.4 Relaties die een entiteit met zichzelf verbinden

Relaties konden tot nu toe twee of meer entiteiten met elkaar verbinden. Soms is het echter handig om een relatie tweemaal met dezelfde entiteit te verbinden. Stel dat we willen aangeven dat een werknemer rapporteert aan een andere werknemer (zijn manager), en dat deze andere werknemer rapportages ontvangt van meerdere werknemers. Dit kan als volgt worden gemodelleerd:



► **Figuur 2.22** Een voorbeeld van een relatie die twee keer is verbonden met dezelfde entiteit, en de manier waarop zo'n relatie gelezen moet worden

Zoals het voorbeeld laat zien, hebben we aan één entiteit genoeg omdat managers ook werknemers zijn en (in dit voorbeeld) dezelfde attributen hebben. Zoals elke relatie kunnen we ook de relatie uit figuur 2.22 in twee richtingen lezen. Het is gebruikelijk om dit soort relaties rechtsonder de entiteit te tekenen, zodat er geen misverstand kan bestaan over de richting waarin we moeten lezen. Hieronder tot slot nog een voorbeeld van een relatie die twee keer met dezelfde entiteit is verbonden.



► **Figuur 2.23** Sommige studenten begeleiden meerdere medestudenten. Alle studenten worden begeleid door één medestudent.

Samenvatting

ERD

Voordat we een database bouwen, is het verstandig om eerst een ontwerp te maken. Het **entiteit-relatie-diagram** (ERD) is een techniek om databases te ontwerpen. Hierbij hebben we drie symbolen tot onze beschikking:

- **entiteit** — iets waarover de organisatie iets wil vastleggen
- **relatie** — een zinvolle samenhang tussen twee entiteiten
- **attribuut** — eigenschappen die van een entiteit of relatie worden vastgelegd

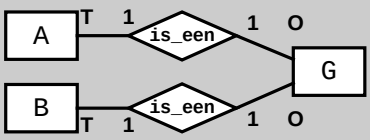
Cardinaliteit

Van een relatie wordt in een ERD ook de **cardinaliteit** vastgelegd. De cardinaliteit bestaat uit **functionaliteit** en **totaliteit**. Met de functionaliteit kan worden aangegeven of een entiteit meerdere keren voor kan komen in relatie tot een andere entiteit (**n**), of maximaal één keer (**1**). Met de totaliteit kan worden aangegeven of een entiteit altijd voorkomt in relatie tot een andere entiteit (**T**), of niet altijd (**0**).

Bijzondere relaties

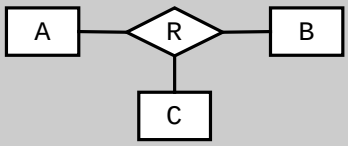
Om een zo efficiënt mogelijke database te ontwerpen, hebben we een aantal bijzondere relaties tot onze beschikking:

■ generalisatie



De overlappende attributen van A en B worden ondergebracht bij G

■ meerdimensionale relatie

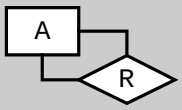


Relatie R verbindt de entiteiten A, B en C met elkaar

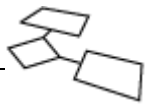
■ specialisatie

Attributen die in A veel lege velden veroorzaken, worden ondergebracht in S

■ tweemaal met dezelfde entiteit



A wordt via relatie R met zichzelf verbonden



3 Van ERD naar relationeel model

3.1 Inleiding

Voordat we aan de hand van het ERD een database kunnen gaan bouwen, moet er nog een aantal stappen worden uitgevoerd: het ERD moet vertaald worden naar het relationele model. Deze vertaling zorgt ervoor dat de juiste sleutels bij de juiste tabel terechtkomen. Immers, in de attributenlijst bij ons ERD hebben we nog geen rekening gehouden met sleutels die, zoals in hoofdstuk 1 is behandeld, verantwoordelijk zijn voor de koppeling van de ene tabel aan de andere.

Nu zouden we zo'n beetje uit de losse pols hier en daar wat sleutels kunnen toewijzen aan tabellen, maar in de praktijk leidt dit vaak tot fouten. Gelukkig is er een stappenplan dat we kunnen volgen bij het vertalen van ERD naar relationeel model. Wanneer we dit stappenplan volgen, zal de vertaling altijd goed gaan (ervan uitgaande dat het ERD in orde is). Het stappenplan bestaat uit *drie basisregels* en tot slot nog een *herstructurering*.

3.2 De basisregels

3.2.1 Basisregel 1: unieke namen geven

basisregel 1

Basisregel 1: *Geef iedere entiteit en relatie een unieke naam.*

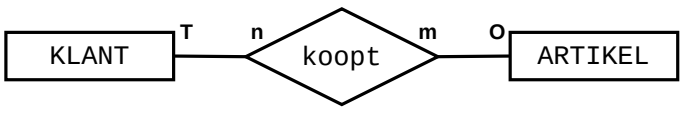
In een database is het niet toegestaan dat twee tabellen dezelfde naam hebben. Daarom moeten we er in ons ERD voor zorgen dat elke entiteit en elke relatie een unieke naam heeft. Alleen de bijzondere relatie met de naam IS_EEN mag meer dan één keer voorkomen (deze kan namelijk nooit een tabel worden). Vaak komen relatienamen als "heeft" en "bevat" meer dan één keer voor. Dit kunnen we het eenvoudigst oplossen door ze te nummeren, dus: "heeft 1", "heeft 2" etc.

3.2.2 Basisregel 2: entiteiten worden tabellen met sleutels

basisregel 2

Basisregel 2: *Maak van iedere entiteit een tabel (met zijn attributen) en geef iedere tabel een sleutel; bij een IS_EEN relatie door overerving.*

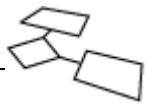
Aan de hand van het ERD maken we nu een lijst met alle entiteiten en hun attributen (als we dat niet al gedaan hadden in de vorm van een attributenlijst). Vanaf nu spreken we niet meer van entiteiten maar van tabellen. Per tabel moet nu gekeken worden of één van de attributen kan functioneren als sleutel. Als geen van de attributen een record uit de tabel uniek maakt, kunnen we twee dingen doen: een samengestelde sleutel gebruiken of een nieuw attribuut toevoegen om als sleutel te laten functioneren.



Attributenlijst

```
KLANT       : naam, adres
ARTIKEL     : artikelcode, omschrijving
koopt       : datum
```

► **Figuur 3.1** Voorbeeld van een ERD



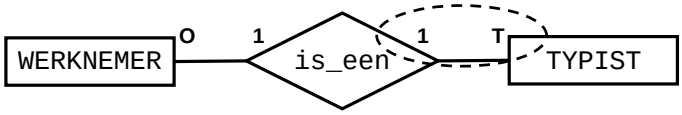
Na het toepassen van basisregel 2 op het ERD van figuur 3.2 krijgen we:

Attributenlijst na basisregel 2
```
KLANT       : klantnummer, naam, adres
ARTIKEL     : artikelcode, omschrijving
```

► **Figuur 3.2** De attributenlijst van het ERD uit figuur 3.2 na het toepassen van basisregel 2

Zoals in figuur 3.2 is te zien, wordt de sleutel van een tabel onderstreept. De entiteit ARTIKEL wordt een tabel, met als sleutel het attribuut ARTIKELCODE. Ook de entiteit KLANT wordt een tabel, maar geen enkel attribuut is geschikt om als sleutel te dienen. Daarom voegen we in deze fase een nieuw attribuut KLANTNUMMER toe, dat als sleutel gaat functioneren.

Een speciaal geval bij het bepalen van de sleutel, is de IS_EEN relatie. De entiteit die aan de 1T-kant verbonden is met zo'n relatie, krijgt haar sleutel van de entiteit aan de andere kant van de relatie. Met 1T-kant wordt de kant met functionaliteit 1 en totaliteit T bedoeld.



Attributenlijst
```
WERKNEMER   : werknemernr, naam, salaris
TYPIST      : typevaardigheid
```

Attributenlijst na basisregel 2
```
WERKNEMER   : werknemernr, naam, salaris
TYPIST      : werknemernr, typevaardigheid
```

► **Figuur 3.3** Voorbeeld van het toepassen van basisregel 2 bij een ERD met een is_een relatie

Zoals het voorbeeld laat zien, wordt de sleutel van de tabel WERKNEMER gewoon het attribuut WERKNEMERNR. De tabel TYPIST is echter een specialisatie van de tabel WERKNEMER (vandaar de IS_EEN relatie en de cardinaliteit 1T) en overerft dus de sleutel daarvan. Daarom is er als sleutel een attribuut WERKNEMERNR in de tabel TYPIST opgenomen. Deze regel geldt voor zowel specialisatie als generalisatie.

3.2.3 Basisregel 3: relaties worden tabellen met sleutels

basisregel 3

Basisregel 3: *Maak van iedere relatie een tabel (met zijn attributen) behalve de IS_EEN relaties. De sleutel ontstaat door toevoeging van de sleutels van de verbindende entiteiten.*

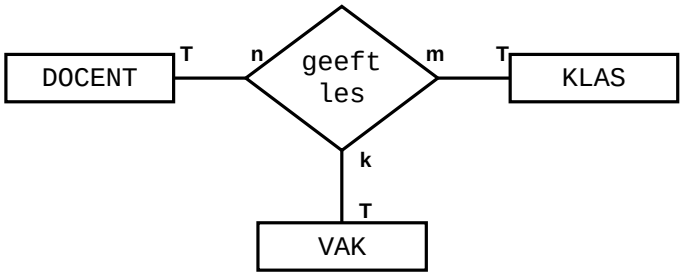
De attributenlijst zoals die er na basisregel 2 uitziet, wordt bij basisregel 3 uitgebreid met een aantal nieuwe tabellen: de relaties uit het ERD. De attributenlijst uit figuur 3.2 zou er na basisregel 3 als volgt uitzien:

Attributenlijst na basisregel 3
```
KLANT       : klantnummer, naam, adres
ARTIKEL     : artikelcode, omschrijving
koopt       : klantnummer, artikelcode, datum
```

► **Figuur 3.4** Attributenlijst van figuur 3.2 na het toepassen van basisregel 3

Er is een nieuwe tabel KOOPT bijgekomen. Het attribuut DATUM was al een attribuut van de relatie KOOPT en keert in deze attributenlijst weer gewoon terug. De sleutel van KOOPT kiezen we niet zelf, deze ontstaat door de sleutels van de entiteiten die met deze relatie zijn verbonden over te nemen. In dit geval zijn dat de entiteiten KLANT en ARTIKEL, met respectievelijk de sleutels KLANTNUMMER en ARTIKELCODE. Samen zijn zij een samengestelde sleutel.

Deze regels worden op exact dezelfde wijze toegepast op meerdimensionale relaties en relaties die tweemaal met dezelfde entiteit zijn verbonden. Onderstaande voorbeelden illustreren dit.



Attributenlijst

```
DOCENT      : naam, salaris
KLAS        : klascode, opmerkingen
VAK         : vakcode, omschrijving
```

Attributenlijst na basisregel 2

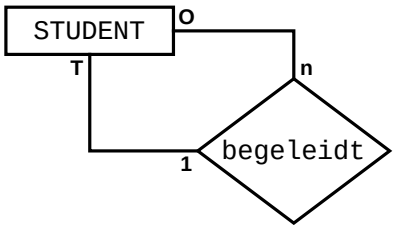
```
DOCENT      : docentnummer, naam, salaris
KLAS        : klascode, opmerkingen
VAK         : vakcode, omschrijving
```

Attributenlijst na basisregel 3

```
DOCENT      : docentnummer, naam, salaris
KLAS        : klascode, opmerkingen
VAK         : vakcode, omschrijving
geeft les   : docentnummer, klascode, vakcode
```

► **Figuur 3.5** De basisregels bij een driedimensionale relatie

Na de derde basisregel is er een tabel GEEFT LES met een sleutel die is samengesteld uit de sleutels van de drie omliggende entiteiten.



Attributenlijst

```
STUDENT     : studentnummer, naam
begeleidt   : begindatum, einddatum
```

Attributenlijst na basisregel 2

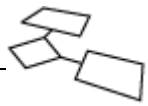
```
STUDENT     : studentnummer, naam
```

Attributenlijst na basisregel 3

```
STUDENT     : studentnummer, naam
begeleidt   : studentnummer, studentnummer,
              begindatum, einddatum
```

► **Figuur 3.6** De basisregels bij een relatie die tweemaal met dezelfde entiteit is verbonden

De tabel BEGELEIDT heeft na basisregel 3 een samengestelde sleutel die bestaat uit twee attributen STUDENTNUMMER. Dit komt natuurlijk doordat de betreffende relatie twee keer met de entiteit STUDENT is verbonden. Voorlopig kunnen we dit zo laten staan, maar wanneer we meer duidelijkheid willen of uiteindelijk de database gaan bouwen, zullen we zelf voor (één van) deze attributen een andere naam moeten kiezen; bijvoorbeeld STUDENTNUMMER en BEGELEIDEND- STUDENTNUMMER.



3.3 Herstructureren

herstructureren

Door de basisregels toe te passen op ons ERD, zorgen we ervoor dat alle entiteiten en relaties tabellen worden met een sleutel. Daarmee zijn we er echter nog niet, want om een goed functionerende relationele database te krijgen, moeten sommige attributen (met name sleutels) nog worden overgeheveld naar andere tabellen. Dit proces noemen we *herstructureren*.

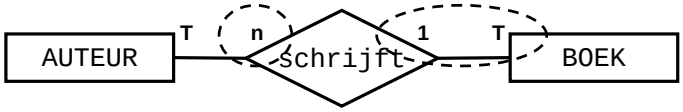
Bij het herstructureren moet goed worden gekeken naar de cardinaliteit uit het ERD. Afhankelijk van deze cardinaliteit worden sleutels wel of niet ondergeschoven bij een andere tabel. We zullen de verschillende mogelijkheden bespreken aan de hand van de drie soorten functionaliteit: n-op-1, n-op-m en 1-op-1.

3.3.1 De n-op-1 relaties

Bij het herstructureren van n-op-1 (of natuurlijk 1-op-n) relaties speelt ook de totaliteit een rol. Hier zullen we de n-op-1T relatie bespreken, in paragraaf 3.3.4 wordt ingegaan op relaties met de totaliteit O (optioneel).

n-op-1T relaties herstructureren

Regel: *Alle attributen van n-op-1T relaties worden ondergeschoven bij de entiteit aan de 1T-kant.*



Attributenlijst

```
AUTEUR      : naam
BOEK        : ISBN, titel
schrijft    : publicatiedatum
```

Attributenlijst na basisregels

```
AUTEUR      : auteurcode, naam
BOEK        : ISBN, titel
schrijft    : auteurcode, ISBN, publicatiedatum
```

Attributenlijst na herstructureren

```
AUTEUR      : auteurcode, naam
BOEK        : ISBN, titel, auteurcode,
              publicatiedatum
```

► **Figuur 3.7** Het herstructureren van een n-op-1T relatie

Bij het herstructureren verdwijnt de tabel SCHRIJFT. De attributen van die tabel (AUTEURCODE, ISBN en PUBLICATIEDATUM) worden namelijk ondergeschoven bij de tabel BOEK. Omdat deze tabel al een attribuut ISBN bevat, nemen we deze niet nog een keer op. De sleutel van de tabel BOEK blijft gewoon het attribuut ISBN, dit was immers al de sleutel en het toevoegen van extra attributen verandert niets aan het feit dat dit attribuut een BOEK-record uniek maakt.

Als we kijken naar de tabel BOEK die zo ontstaat, zien we dat deze tabel eigenlijk heel logisch in elkaar zit. De publicatiedatum kan prima worden opgenomen in de tabel BOEK, net als de auteurcode. Door de relatie SCHRIJFT in ons ERD is er nu een verwijzing ontstaan van de tabel BOEK naar de tabel AUTEUR met behulp van het attribuut AUTEURCODE.

AUTEUR

Auteurcode	Naam
A23	Annie
…	…

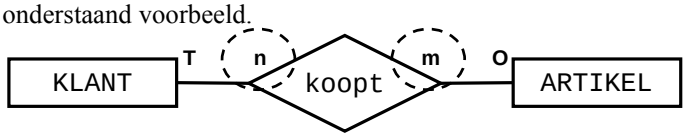
BOEK

ISBN	Titel	Auteurcode	Pub.datum
902671480	Piet Puk	A23	12-06-2002
…	…	…	…

► **Figuur 3.8** Op basis van het ERD uit figuur 3.7 worden in de database deze tabellen gemaakt

3.3.2 De n-op-m relaties

Wanneer we zouden proberen de sleutels van een n-op-m relatie, net als bij een n-op-1T relatie, onder te schuiven, komen we in de problemen. Bekijk onderstaand voorbeeld.



Attributenlijst
KLANT : klantnr, naam
ARTIKEL : artikelnr, omschrijving

Attributenlijst na basisregels
KLANT : klantnr, naam
ARTIKEL : artikelnr, omschrijving
koopt : klantnr, artikelnr

► **Figuur 3.9** Voorbeeld van een ERD met een n-op-m relatie

Bij het herstructureren van bovenstaand ERD zouden we er voor kunnen kiezen om de relatie KOOPT onder te schuiven bij de entiteit KLANT of bij de entiteit ARTIKEL. De tabellen hieronder tonen aan dat deze keuze zou resulteren in redundantie en daarmee ingaat tegen het principe van het relationele model.

- Wanneer wordt ondergeschoven bij KLANT:

KLANT

Klantnr	Naam	Artikelnr
10	Piet	X8
10	Piet	X9
…	…	…

- Of wanneer wordt ondergeschoven bij ARTIKEL:

ARTIKEL

Artikelnr	Omschrijving	Klantnr
X8	Raketmotor	10
X8	Raketmotor	2
…	…	…

► **Figuur 3.10** De problemen die ontstaan bij het onderschuiven van een n-op-m relatie

koppeltabel
aggregaat

Daarom zit er bij een n-op-m relatie maar één ding op: zij wordt een zelfstandige tabel. Zo'n tabel die ontstaat uit een n-op-m relatie, koppelt in feite de twee verbindende tabellen aan elkaar en wordt daarom ook wel een *koppeltabel* of *aggregaat* genoemd. De juiste oplossing voor het ERD van figuur 3.9 wordt hieronder gegeven.

n-op-m relaties herstructureren

Regel: *N-op-m relaties worden een zelfstandige tabel.*

Attributenlijst na herstructureren
KLANT : klantnr, naam
ARTIKEL : artikelnr, omschrijving
koopt : klantnr, artikelnr

- De tabellen zoals die in de database zullen worden gemaakt:

KLANT

Klantnr	Naam
2	Arie
10	Piet
89	Klaas
…	…

KOOPT

Klantnr	Artikelnr
10	X8
10	A4
2	X8
…	…

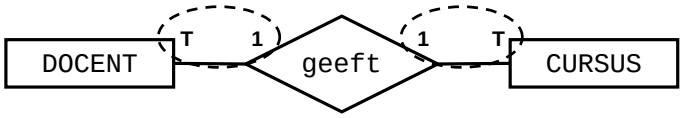
ARTIKEL

Artikelnr	Omschrijving
A4	KH-11 Satelliet
X8	Raketmotor
X9	F-22 Raptor
…	…

► **Figuur 3.11** Figuur 3.9 geherstructureerd en de tabellen zoals die in de database gemaakt worden

3.3.3 De 1-op-1 relaties

Net als bij de n-op-1 relaties, speelt ook bij 1-op-1 relaties de totaliteit een rol. Hoe moet worden omgegaan met relaties met de totaliteit O, wordt behandeld in paragraaf 3.3.4. In deze paragraaf behandelen we de T1-op-1T relatie. Hieronder staat een voorbeeld van zo'n relatie.



Attributenlijst
DOCENT : docentnr, docentnaam
CURSUS : cursuscode, omschrijving

► **Figuur 3.12** Een voorbeeld van een ERD met een T1-op1T relatie

In het ERD van figuur 3.12 geven docenten altijd één cursus, en wordt een bepaalde cursus altijd door één docent gegeven. Hieruit volgt dat het geen enkel probleem zal opleveren wanneer we alle gegevens van docenten en cursussen in één tabel bewaren. De attributen van de entiteiten DOCENT en CURSUS kunnen bij het herstructureren dus worden samengevoegd in één tabel, zoals hieronder in figuur 3.13 is weergegeven. De naam van de gezamenlijke tabel mogen we zelf kiezen, evenals de sleutel (DOCENTNR of CURSUSCODE).

T1-op-1T relaties herstructureren

Regel: *Bij een T1-op-1T relatie kunnen de attributen van de verbindende entiteiten en de relatie worden samengevoegd in één tabel.*

Attributenlijst na herstructureren
DOCENT_CURSUS: docentnr, docentnaam, cursuscode, omschrijving

- De tabel die in de database zal worden gemaakt:

DOCENT_CURSUS

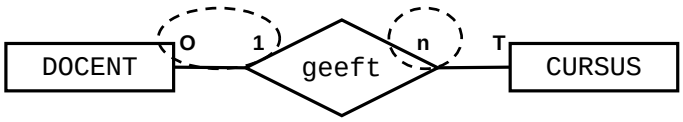
Docentnr	Docentnaam	Cursuscode	Omschrijving
D20	Jan	IK03	Gegevensmodellering
D21	Marieke	CSB1	Cursus voor beginners
…	…	…	…

► **Figuur 3.13** Het ERD van figuur 3.12 geherstructureerd en de tabel zoals die in de database zal worden gemaakt

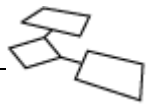
Op deze manier herstructureren is mogelijk, maar niet verplicht. De ontwerper moet op basis van de situatie in de betreffende organisatie, rekening houdend met veranderingen in die situatie in de toekomst, zelf beslissen of het samenvoegen van twee entiteiten in één tabel verstandig is.

3.3.4 Relaties met totaliteit O

Tot nu toe hebben we alleen het herstructureren van relaties met de totaliteit T (totaal) behandeld. Wanneer we bij het herstructureren attributen van relaties onder zouden schuiven bij een entiteit met de totaliteit O (optioneel), levert dit een probleem op. In de betreffende tabel zullen lege velden ontstaan, want de relatie is per slot van rekening optioneel en hoeft dus niet altijd te worden ingevuld.



Attributenlijst
DOCENT : naam
CURSUS : cursuscode, omschrijving



Attributenlijst na basisregels

```
DOCENT      : docentnr, naam
CURSUS      : cursuscode, omschrijving
geeft       : docentnr, cursuscode
```

► **Figuur 3.14** Voorbeeld van een ERD met een O1-op-n relatie

Er is in figuur 3.14 sprake van een 1-op-n relatie, maar dit betekent niet dat de attributen van de relatie GEEFT zonder meer mogen worden ondergeschoven aan de 1-kant bij de entiteit DOCENT. Dit zou namelijk resulteren in de volgende tabel DOCENT:

DOCENT

Docentnr	Naam	Cursuscode
D20	Pieter	
D21	Arie	
D22	Annie	
D23	Marieke	CSB1
D24	Piet	
…	…	…

Voor elke docent die geen cursus geeft, zal het veld CURSUSCODE leeg blijven

► **Figuur 3.15** Door de attributen van een relatie onder te schuiven aan de kant met de totaliteit O ontstaan lege velden in de database

Zoals we in hoofdstuk 2 al konden zien, zijn dergelijke constructies met veel lege velden niet efficiënt en dus niet gewenst. Daarom houden we de volgende regel aan:

relaties met totaliteit O herstructureren

Regel: *De attributen van een relatie worden in principe nooit ondergeschoven aan een kant met de totaliteit O.*

Deze regel geldt voor alle relaties, dus ook voor meerdimensionale relaties, relaties die tweemaal met dezelfde entiteit zijn verbonden en O1-op-1T relaties (die kunnen natuurlijk wel worden ondergeschoven aan de 1T-kant). De meest efficiënte oplossing voor het ERD van figuur 3.14, zou zijn om van de relatie GEEFT een zelfstandige tabel te maken:

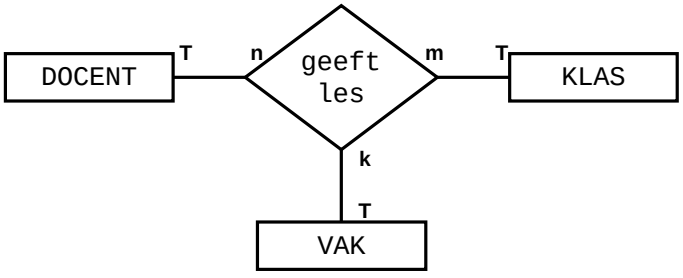
Attributenlijst na herstructureren

```
DOCENT      : docentnr, naam
CURSUS      : cursuscode, omschrijving
geeft       : docentnr, cursuscode
```

► **Figuur 3.16** Een O1-op-n relatie kan het best een zelfstandige tabel worden

3.3.5 Bijzondere relaties herstructureren

De regels voor het herstructureren zijn ook van toepassing op meerdimensionale relaties en relaties die meerdere malen met dezelfde entiteit zijn verbonden, zoals onderstaande voorbeelden laten zien.



Attributenlijst

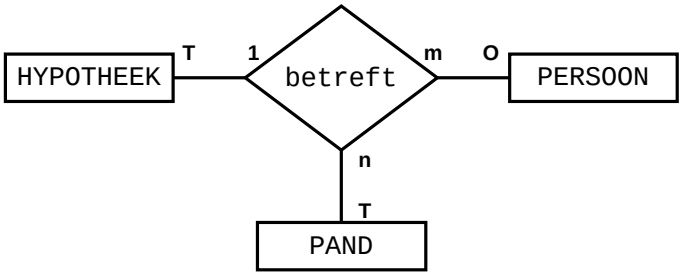
```
DOCENT      : naam, salaris
KLAS        : klascode, opmerkingen
VAK         : vakcode, omschrijving
```

Attributenlijst na basisregels

DOCENT : docentnummer, naam, salaris
KLAS : klascode, opmerkingen
VAK : vakcode, omschrijving
geeft les : docentnummer, klascode, vakcode

Attributenlijst na herstructureren

DOCENT : docentnummer, naam, salaris
KLAS : klascode, opmerkingen
VAK : vakcode, omschrijving
geeft les : docentnummer, klascode, vakcode



Attributenlijst

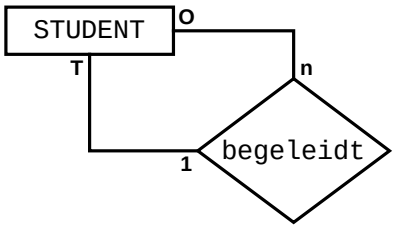
HYPOTHEEK : bedrag, afsluitdatum
PERSOON : sofinr, naam, telefoon
PAND : postcode, huisnr, vraagprijs

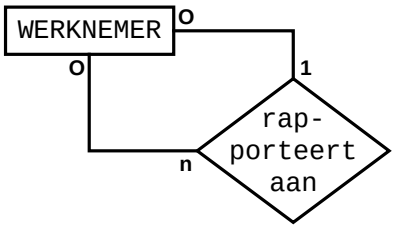
Attributenlijst na basisregels

HYPOTHEEK : hypotheeknr, bedrag, afsluitdatum
PERSOON : sofinr, naam, telefoon
PAND : postcode, huisnr, vraagprijs
betreft : hypotheeknr, sofinr, postcode, huisnr

Attributenlijst na herstructureren

HYPOTHEEK : hypotheeknr, bedrag, afsluitdatum, sofinr, postcode, huisnr
PERSOON : sofinr, naam, telefoon
PAND : postcode, huisnr, vraagprijs





Attributenlijst

STUDENT : studentnr,naam
begeleidt : begindatum

Attributenlijst

WERKNEMER : naam

Attributenlijst na basisregels

STUDENT : studentnr,naam
begeleidt : studentnr, studentnr[1], begindatum

Attributenlijst na basisregels

WERKNEMER : wnr, naam
rapp aan : wnr, wnr[2]

Attributenlijst na herstructureren

STUDENT : studentnr,naam, studentnr[1], begindatum

Attributenlijst na herstructureren

WERKNEMER : wnr, naam
rapp aan : wnr, wnr[2]

[1] Dit attribuut is het studentnummer van de begeleidende student
[2] Dit attribuut is het werknemernummer van de werknemer die rapportages ontvangt

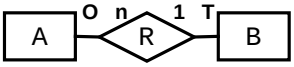
► **Figuur 3.17** Enkele voorbeelden van het herstructureren van bijzondere relaties

Samenvatting

Van ERD naar database

Het ERD moet vertaald worden naar het relationeel model voordat we daadwerkelijk een database kunnen bouwen. Het complete proces:

■ 1. ERD maken



Attributenlijst
A: AS
B: BS
R: R1

■ 2. Vertalen

Attributenlijst na basisregels
A: AS
B: BS
R: AS, BS, R1

Attributenlijst na herstructureren
A: AS
B: BS, AS, R1

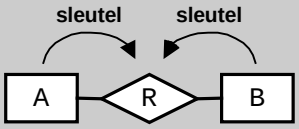
■ 3. Bouwen

Database bouwen, bijvoorbeeld met Microsoft Access

Vertaling naar relationeel model

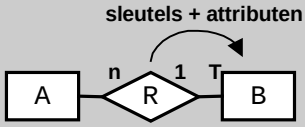
De vertaling naar het relationele model gebeurt in twee stappen: het toepassen van de **basisregels** en **herstructureren**. In grote lijnen gebeurt er het volgende:

■ 1. basisregels



De sleutel van elke entiteit wordt bepaald. Relaties ontvangen hun sleutel van de verbindende entiteiten

■ 2. herstructureren



Afhankelijk van de cardinaliteit worden sleutels en attributen van sommige relaties ondergeschoven bij een entiteit

In bovenstaand voorbeeld wordt (in twee stappen) de sleutel van A via R ook opgenomen in B, waar deze als verwijzing naar A zal functioneren.

Basisregels

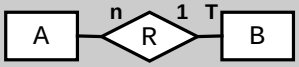
Eerste stap bij het vertalen naar het relationele model is het toepassen van de drie basisregels:

1. Geef iedere entiteit en relatie een unieke naam
2. Maak van iedere entiteit een tabel (met zijn attributen) en geef iedere tabel een sleutel (bij IS_EEN door overerving)
3. Maak van iedere relatie een tabel (met zijn attributen) behalve IS_EEN. De sleutel ontstaat door toevoeging van de sleutels van de verbindende entiteiten

Herstructureren

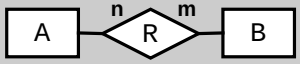
Na het toepassen van de basisregels moet worden geherstructureerd. Daarbij moet worden gekeken naar de functionaliteit en totaliteit:

■ n-op-1T relaties



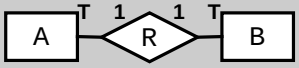
Alle attributen van relatie R worden ondergeschoven bij entiteit B

■ n-op-m relaties



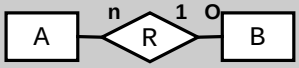
De relatie R wordt een zelfstandige tabel

■ T1-op-1T relaties



Alle attributen van A, B en R kunnen worden samengevoegd in één tabel

■ relaties met totaliteit O



Attributen worden niet ondergeschoven aan een kant met de totaliteit O

4 Een ERD ontwerpen

4.1 Inleiding

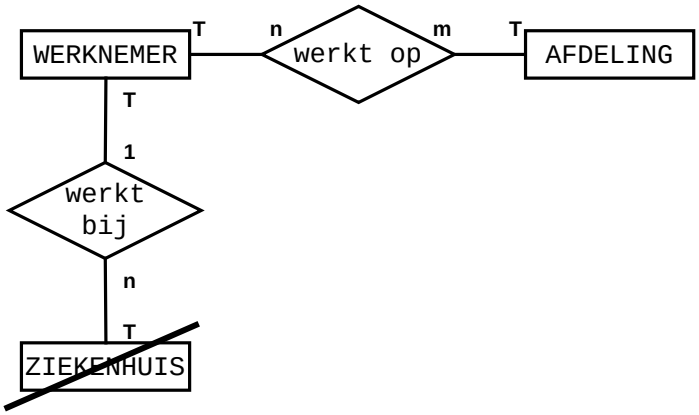
In hoofdstuk 2 is uitgelegd hoe we een ERD kunnen maken en aan welke regels we ons daarbij moeten houden. In hoofdstuk 3 werd vervolgens beschreven hoe we van een ERD kunnen komen tot een relationeel model, op basis waarvan de database gebouwd kan worden. Tijdens het ontwerpproces komen we vaak voor belangrijke keuzes te staan die een groot effect hebben op de database die we er uiteindelijk mee gaan bouwen. Een aantal van deze keuzes en een aantal valkuilen bij het ontwerpen van een ERD, zullen in dit hoofdstuk aan bod komen.

4.2 Relevantie

Bij het ontwerpen van een ERD moeten we altijd goed voor ogen blijven houden wat het doel van het ontwerp is: uiteindelijk willen we een efficiënte database bouwen die voldoet aan de eisen van de organisatie. De gegevens die nodig zijn om de organisatie goed te kunnen laten functioneren zullen we in de database moeten kunnen terugvinden, niks meer en niks minder. Daarom moeten we er op letten dat we alleen relevante entiteiten, relaties en attributen opnemen in ons ERD.

4.2.1 Relevantie van entiteiten

Wanneer we bepalen welke entiteiten er nodig zijn, lopen we het risico om er te weinig of te veel op te nemen. Onderstaand voorbeeld is een ontwerp voor het personeelsbestand van een ziekenhuis.

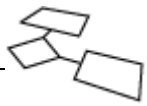


► **Figuur 4.1** Een ERD voor het personeelsbestand van een ziekenhuis met een irrelevante entiteit

De entiteit ZIEKENHUIS is in dit ERD irrelevant. De eenvoudigste manier om dit duidelijk te maken is door ons voor te stellen hoe deze tabel er straks in de database uit zal gaan zien. Wat voor records zullen er in komen te staan? Wanneer we dit doen blijkt al gauw dat er altijd maar één record in deze tabel zal komen te staan, gevuld met de gegevens van het ziekenhuis waar we het ontwerp voor maken.

irrelevante entiteiten

Regel: *Entiteiten die een tabel worden die altijd één record zal bevatten, zijn irrelevant en worden niet opgenomen in een ERD.*



De entiteit ZIEKENHUIS uit figuur 4.1 zou er als tabel zo uit komen te zien:

ZIEKENHUIS

Naam	Adres	Plaats
Medisch Centrum Blikdorp	Dorpsstraat 8	Blikdorp

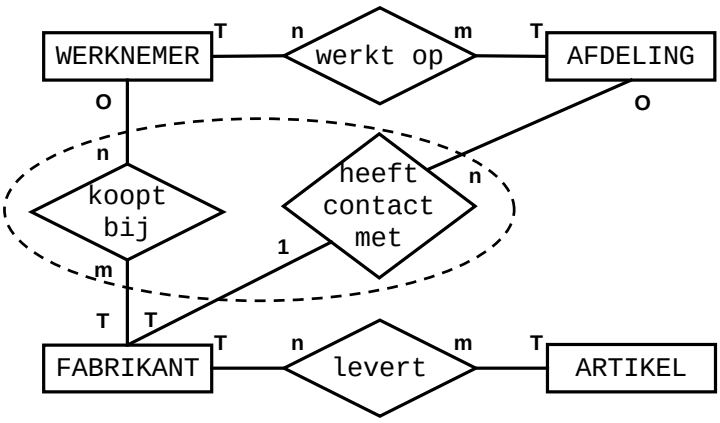
► **Figuur 4.2** Een (nutteloze) tabel die altijd één record bevat, met daarin wat gegevens over de eigen organisatie

4.2.2 Relevantie van relaties

In de vorige hoofdstukken hebben we kunnen zien dat het leggen van een relatie tussen entiteiten belangrijke gevolgen heeft voor de structuur van de database. Het zorgt er onder meer voor dat er met behulp van sleutels verwijzingen tussen de tabellen komen. Het ontwerpen van een ERD is dus niet een kwestie van eerst de entiteiten bepalen, om daar dan vervolgens op gevoel hier en daar wat relaties tussen te tekenen. Het is zaak dat we alleen op die plaatsen relaties leggen, waar we ook echt een relatie willen bijhouden. Lees nu onderstaande casus, die dient als basis voor het ontwerp van een database voor een handelsonderneming.

> Werknemers kunnen in de loop der tijd op meerdere afdelingen werken en werken altijd op een bepaalde afdeling. Op een afdeling werken altijd meerdere weknemers. Een aantal afdelingen verkoopt artikelen. Alle artikelen kunnen worden geleverd door meerdere fabrikanten en worden verkocht door meerdere afdelingen. Een fabrikant kan meerdere soorten artikelen leveren en wordt pas in het fabrikantenbestand opgenomen wanneer er voor het eerst artikelen bij hem worden besteld. Men wil graag bijhouden welke fabrikant welke artikelen levert.

De benodigde entiteiten zijn alvast onderstreept: WERKNEMER, AFDELING, ARTIKEL en FABRIKANT. Het bepalen van de entiteiten is in deze casus echter niet het moeilijkst, lastiger is om er de juiste relaties tussen te leggen. Bekijk onderstaande uitwerking van de casus.



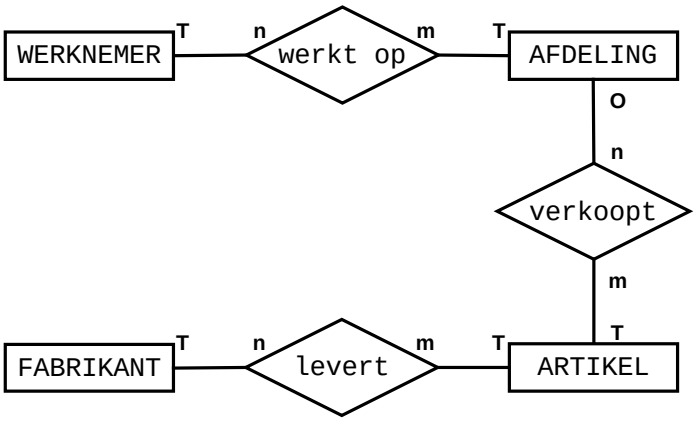
► **Figuur 4.3** Een (foutieve) uitwerking van de casus

In figuur 4.3 zijn de juiste entiteiten opgenomen, maar kloppen de relaties niet met de situatie in de organisatie. Uit de casusbeschrijving blijkt dat men alleen geïnteresseerd is in de relaties tussen de entiteiten:

- WERKNEMER en AFDELING
- AFDELING en ARTIKEL
- ARTIKEL en FABRIKANT

De relaties KOOPT BIJ en HEEFT CONTANCT MET in figuur 4.3 mogen er dan wel logisch uitzien, voor deze casus zijn ze irrelevant. Er zijn ongetwijfeld organisaties waarvoor de gegeven uitwerking prima voldoet, maar voor de organisatie uit de casus zal het een verkeerde database opleveren. Men is helemaal niet geïnteresseerd in de contacten die elke afdeling heeft met de fabrikanten en wil daar in de database dus ook geen gegevens over bijhouden. Wel is men bijvoorbeeld geïnteresseerd in welke artikelen door welke afdeling

verkocht worden, maar die gegevens zullen met bovenstaand ontwerp niet in de database kunnen worden teruggevonden. Het juiste ontwerp bij de casus:



► **Figuur 4.4** De juiste uitwerking van de casus

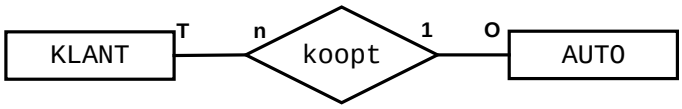
In dit ERD zijn de genoemde relaties allemaal terug te vinden. In de database zal daarom kunnen worden teruggevonden:

- op welke afdelingen een werknemer heeft gewerkt (dankzij de relatie WERKT OP)
- welke artikelen elke afdeling verkoopt (dankzij de relatie VERKOOPT)
- welke fabrikant welke artikelen levert (dankzij de relatie LEVERT)

4.3 Nogmaals: attributen aan relaties

datum gegevens
numerieke gegevens

Bij attributen aan entiteiten kunnen we ons meestal gemakkelijk iets voorstellen: van een entiteit KLANT willen we bijvoorbeeld vaak een naam, adres, postcode etc. bijhouden. Maar wanneer hebben relaties nu eigenlijk attributen? En wat voor soort attributen zijn dat dan? Om met de laatste vraag te beginnen: attributen van relaties betreffen bijna altijd *datumgegevens* of *numerieke gegevens* (gegevens waarmee kan worden gerekend, zoals een prijs of een aantal). Ze vertellen meestal iets over het wanneer of over het hoeveel van de relatie. Hieronder is een ERD voor een autodealer gegeven.



Attributenlijst

```
KLANT       : klantnr, naam, adres
AUTO        : chassisnr, merk, type, adviesprijs
koopt       : verkoopdatum, verkoopprijs
```

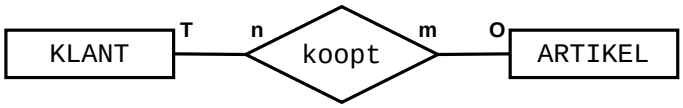
► **Figuur 4.5** Een ERD voor een autodealer

De gegevens die we van de klanten en van de auto's willen bijhouden, spreken voor zich. Maar de prijs waartegen de auto verkocht wordt en de datum waarop dit gebeurt, horen eigenlijk noch bij de klant, noch bij de auto. Daarom zijn dit typisch attributen die we bij een relatie opnemen: ze vertellen iets over de koppeling tussen de entiteiten, en niet direct iets over de entiteiten zelf.

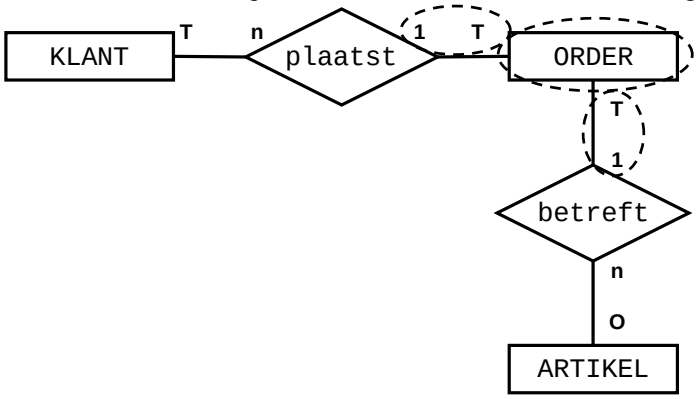
4.4 Nogmaals: n-op-m relaties

Relaties met de functionaliteit n-op-m worden uiteindelijk zelfstandige tabellen in de database. We kunnen er ook voor kiezen om zo'n (koppel)tabel meteen al in ons ERD op te nemen. Figuur 4.6 laat zien dat de entiteit die zo ontstaat, wordt omringd door T1-relaties, waardoor bij herstructureren dezelfde database ontstaat als wanneer we een n-op-m relatie hadden gebruikt. Beide

varianten hebben hun voordeel: de n-op-m relatie ziet er overzichtelijker uit, bij de tweede variant kan een aparte entiteit ORDER worden opgenomen, wat misschien beter aansluit bij de werkwijze van de betreffende organisatie.



- Bovenstaande n-op-m relatie kan eventueel worden vervangen door:

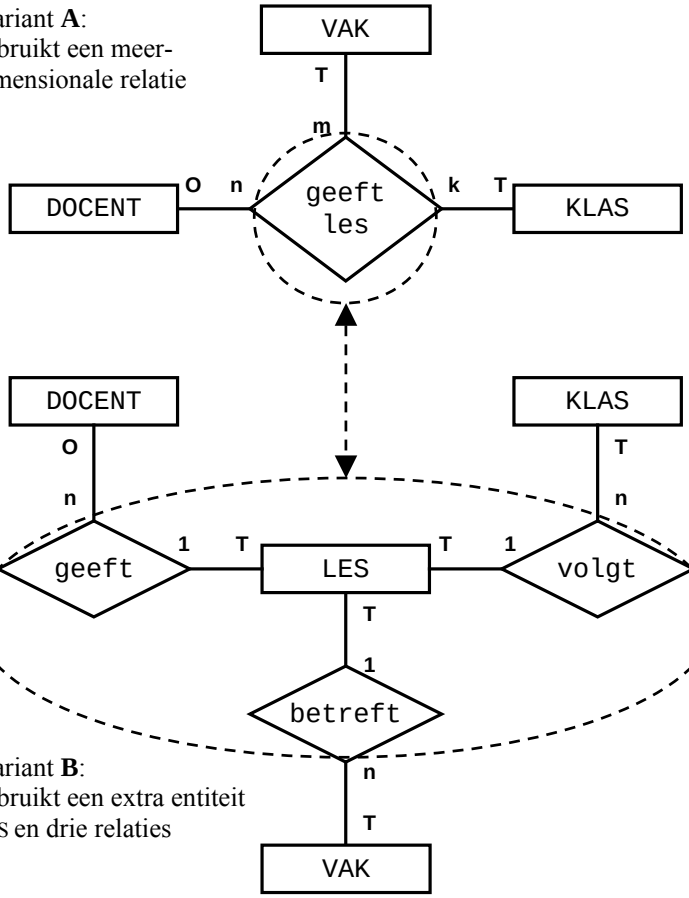


► **Figuur 4.6** De n-op-m relatie koopt wordt vervangen door de entiteit order

4.5 Nogmaals: meerdimensionale relaties

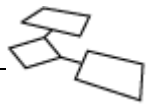
In de praktijk blijkt het vaak lastig om te bepalen wanneer nu wel en wanneer niet een meerdimensionale relatie kan worden gebruikt. Om te begrijpen wat een meerdimensionale relatie precies inhoudt, is het goed om te beseffen dat we in plaats van zo'n relatie, net als bij een n-op-m relatie, ook een extra entiteit kunnen gebruiken. Hieronder zijn twee ER-diagrammen gegeven die er op het eerste gezicht heel verschillend uitzien, maar eigenlijk hetzelfde zijn.

- Variant **A**: gebruikt een meer-dimensionale relatie



- Variant **B**: gebruikt een extra entiteit LES en drie relaties

► **Figuur 4.7** Twee verschillende ER-diagrammen die dezelfde database op zullen leveren



Zowel variant A als variant B zou een goede oplossing zijn, hoewel variant A er natuurlijk een stuk overzichtelijker uitziet. En dat is ook meteen de reden waarom we zouden kiezen voor een meerdimensionale relatie in plaats van een extra entiteit: het is overzichtelijker. Voor de werking van de database maakt het verder niets uit wat we kiezen, want de entiteit LES in variant B is omringd met `T1`-relaties. Dit betekent dat bij het herstructureren van variant B de attributen van de relaties GEEFT, VOLGT en BETREFT allemaal worden ondergeschoven bij de entiteit LES. Zo ontstaan dan precies dezelfde tabellen als wanneer we hadden gekozen voor variant A.

Het is ook belangrijk om ons te realiseren dat een meerdimensionale relatie, net als een gewone tweedimensionale relatie, is bedoeld om entiteiten met elkaar te verbinden. We willen niet alleen gegevens over de entiteiten bijhouden, maar ook welk record uit de ene entiteit is gekoppeld aan welk record uit de andere. De volgende attributenlijst (en tabel) hoort bij variant A uit figuur 4.7:

Attributenlijst

```
DOCENT       : docentnr, naam
KLAS         : klascode, opmerkingen
VAK          : vakcode, omschrijving
geeft les    : lesdatum
```

- Na basisregels en herstructureren zal de relatie GEEFT LES een tabel worden in de database, zoals hieronder is weergegeven (ga dit zelf na):

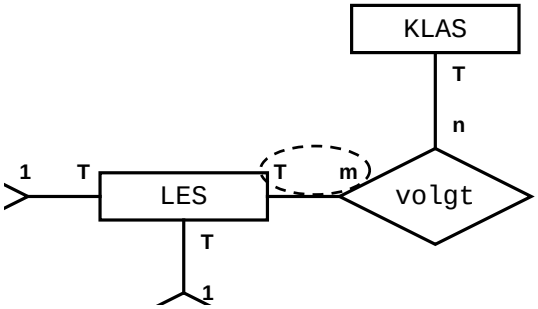
GEEFT LES

Docentnr	Klascode	Vakcode	Lesdatum
D20	1i3	IK13	23-04-2002
D21	1i2	BA12	01-12-2002
…	…	…	…

► **Figuur 4.8** Hoe de meerdimensionale relatie van variant A uiteindelijk een tabel wordt die drie andere tabellen aan elkaar koppelt

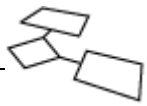
De tabel GEEFT LES zal een samengestelde sleutel bevatten, bestaande uit DOCENTNR, KLASCODE en VAKCODE. Doordat de tabel deze drie sleutels bevat, koppelt hij dus de tabellen DOCENT, KLAS en VAK aan elkaar. Wanneer het een historische database moet worden, zal ook het attribuut LESDATUM in de sleutel moeten worden opgenomen. Anders zou het niet mogelijk zijn dat een bepaalde docent in de loop der jaren meerdere keren hetzelfde vak aan dezelfde klas geeft (wat natuurlijk heel gebruikelijk is binnen een school).

Een meerdimensionale relatie kan echter niet altijd zomaar worden gebruikt. Wanneer variant B uit het ERD van figuur 4.7 ook wordt gebruikt om hoorcolleges (die aan meerdere klassen tegelijk worden gegeven) te registreren, ontstaat er een probleem: LES wordt niet meer omringd door alleen `T1`-relaties.



► **Figuur 4.9** Een bepaalde les wordt aan één of meer klassen tegelijk gegeven

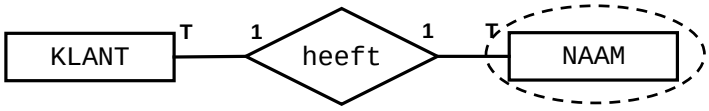
Wanneer we nu een meerdimensionale relatie zouden gebruiken (variant A in figuur 4.7), ontstaan er problemen in de database. De attributen en sleutels van alle relaties die zijn verbonden met de entiteit LES (waaronder VOLGT), zullen dan bij LES worden ondergebracht. Bij de `T1`-relaties gaat dit goed, bij de `Tm`-relatie ontstaat redundantie (zie ook paragraaf 3.3.2).



4.6 Valkuilen

4.6.1 Entiteiten die eigenlijk attributen zijn

Bij het bepalen van de benodigde entiteiten kan het gebeuren dat we een attribuut aanzien voor een entiteit. Onderstaand ERD is een voorbeeld van zo'n situatie.



► **Figuur 4.10** Een attribuut dat wordt gemodelleerd als een entiteit

Een dergelijke constructie is niet echt verkeerd, omdat bij het herstructureren NAAM wordt ondergeschoven bij KLANT, maar het maakt het ERD er niet duidelijker op. Bovendien is het verwarrend, want welke attributen heeft de entiteit NAAM in dit voorbeeld?

4.6.2 Attributen die eigenlijk entiteiten zijn

Andersom komt het wel eens voor dat we per ongeluk een attribuut opnemen dat eigenlijk een entiteit moet zijn. Als we aan zo'n situatie niets doen ontstaat er redundantie in de database. Bekijk onderstaand voorbeeld.

Attributenlijst

```
ARTIKEL: artikelcode, omschrijving,
         prijs, artikelsoort
```

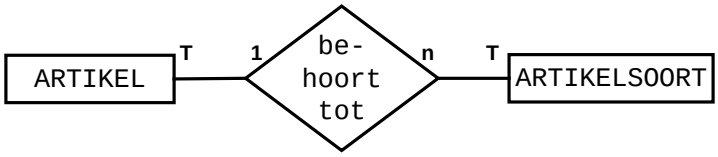
- Na basisregels en herstructureren resulteert dit ERD in de volgende tabel:

ARTIKEL

Artikelcode	Omschrijving	Prijs	Artikelsoort
2508	Spinazie	€ 2,25	groente
2509	Sinasappel	€ 0,75	fruit
3651	Banaan	€ 2	fruit
2921	Appel	€ 1,50	fruit
…	…	…	…

► **Figuur 4.11** Het attribuut artikelsoort veroorzaakt redundantie en had dus eigenlijk een entiteit moeten zijn

Het attribuut ARTIKELSOORT in bovenstaand voorbeeld veroorzaakt redundantie. Dit betekent dat we een ontwerpfout hebben gemaakt: ARTIKELSOORT is geen attribuut maar een entiteit. Het ERD van figuur 4.10 had er dan ook eigenlijk zo uit moeten zien:



Attributenlijst

```
ARTIKEL      : artikelcode, omschrijving, prijs
ARTIKELSOORT : soortcode, soortnaam
```

Attributenlijst na herstructureren

```
ARTIKEL      : artikelcode, omschrijving, prijs,
               soortcode
ARTIKELSOORT : soortcode, soortnaam
```

► **Figuur 4.12** Een verbetering van het ERD van figuur 4.10

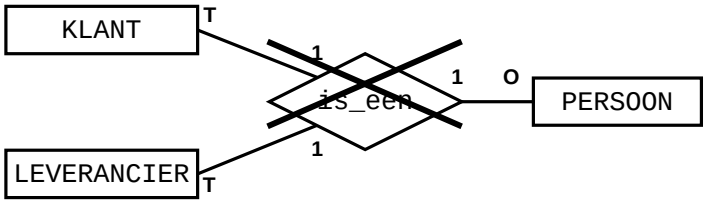
4.6.3 De relatie is_een

regels is_een

Bij het werken met de speciale relatie met de naam IS_EEN geldt een aantal regels. Een IS_EEN relatie:

- moet de naam IS_EEN hebben
- mag alleen worden gebruikt bij generalisatie en specialisatie
- heeft geen attributen
- is altijd tweedimensionaal
- heeft altijd de functionaliteit 1-op-1
- heeft altijd de totaliteit T-op-O
  - bij specialisatie: de T aan de kant van de specialisatie
  - bij generalisatie: de O aan de kant van de generalisatie

Deze regels gelden altijd en de constructie in figuur 4.12 is dan ook fout. Dit is eigenlijk ook logisch omdat deze constructie suggereert dat een KLANT naast een PERSOON ook een LEVERANCIER is, wat natuurlijk nooit de bedoeling van dit ontwerp kan zijn.



► **Figuur 4.13** Een is_een relatie is altijd tweedimensionaal, dit ERD is dus fout

4.6.4 Attributenlijsten

regels attributenlijst

Bij elk ERD moeten de attributen per entiteit worden vermeld. Wanneer we er voor kiezen om de attributen van het ERD niet als ellipsen te tekenen omdat dit onoverzichtelijk wordt, maken we een attributenlijst. In de attributenlijst geven we per entiteit en relatie een opsomming van de gegevens die we er over willen bijhouden: de attributen. Verder doen we in deze fase nog niks met de attributen. In de attributenlijst bij het ERD:

- mogen geen sleutels worden aangegeven
- mogen geen basisregels worden toegepast
- mag niet worden geherstructureerd

Bovenstaande stappen worden in een later stadium bij de basisregels en het herstructureren pas uitgevoerd. Het is belangrijk tussen de attributenlijst bij het ERD en de attributenlijsten die ontstaan na basisregels en herstructureren een duidelijk onderscheid te maken, omdat anders de kans op fouten erg groot is.

4.6.5 Procesgegevens

procesgegeven

Procesgegevens zijn gegevens die het resultaat zijn van een berekening. Denk hierbij bijvoorbeeld aan “het totaalbedrag van een order” of “het aantal klanten.” In een database worden deze gegevens normaal gesproken niet opgeslagen omdat dat risico’s met zich meebrengt: telkens wanneer er ergens gegevens worden veranderd moeten de berekeningen opnieuw worden gemaakt om de procesgegevens bij te werken. Wanneer we (als programmeur) ergens vergeten zo’n berekening uit te voeren, zijn de betreffende procesgegevens inconsistent met alle gevolgen voor de betrouwbaarheid van de database van dien. In de attributenlijst mogen dan ook geen procesgegevens worden opgenomen.

Samenvatting

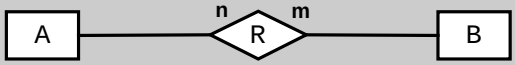
Relevantie

Bij het ontwerpen van een ERD moeten we er op letten dat we alleen relevante entiteiten en relaties opnemen. Een hulpmiddel om **irrelevante entiteiten** te ontdekken, is om ons alvast voor te stellen wat voor records er in die entiteit zullen verschijnen wanneer deze een tabel is geworden. Bevat zo'n tabel altijd maar één record of zinloze gegevens dan is de entiteit niet relevant.
Bij **relaties** moet vooral goed worden gekeken naar de situatie in de organisatie: in welke verbanden tussen entiteiten is men geïnteresseerd? Alleen de relaties waar de organisatie behoefte aan heeft worden in het ERD opgenomen.

Verdieping

Als aanvulling op de theorie valt het volgende op te merken:

- Attributen van relaties zijn meestal **datumgegevens** of **numerieke gegevens**
- `N`-op-`m` relaties (ook meerdimensionale) kunnen eventueel vervangen worden door een entiteit, dus:



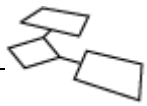
resulteert in dezelfde database als:

- Een meerdimensionale relatie gebruiken we wanneer we geïnteresseerd zijn in de gezamenlijke koppeling van meer dan twee entiteiten
- Een meerdimensionale relatie kan alleen worden gebruikt wanneer elk record uit die relatie altijd maximaal één keer (`T1`) voorkomt in relatie tot de verbindende entiteiten

Valkuilen

Bij het ontwerpen van een ERD moet op de volgende zaken worden gelet:

- soms modelleren we attributen per ongeluk als entiteiten
- soms modelleren we entiteiten per ongeluk als attributen
- bij het gebruiken van de relatie IS_EEN moeten we met een aantal regels rekening houden
- in de attributenlijst van een ERD mogen we nog geen sleutels aangeven en nog niet herstructureren
- in een attributenlijst worden geen **procesgegevens** opgenomen



Register

Opdrachten

Opdrachten

De enige manier om ER-diagrammen te leren maken, is door het te doen. Daarom zijn, naast de theorie, deze opdrachten opgenomen in de syllabus. De opdrachten volgen de opbouw van de theorie: eerst alleen entiteiten, relaties en attributen, vervolgens uitgebreid met cardinaliteit, bijzondere relaties en vertaling naar relationeel model. Bij alle opdrachten wordt uitgegaan van een historische database, tenzij anders vermeld.

- **Opdracht 1**

In een bibliotheek heeft men besloten om alle gegevens over de uitleningen in één grote tabel te bewaren. Hieronder een aantal records uit die tabel.

UITLENINGEN

ISBN	Titel	Auteur	Uitgeleend aan	Datum
930564445	De Romeinen	P. Halleweg	A. Jannes	01-02-2002
930564445	De Romeinen	P. Halleweg	J.J.S. Dokal	23-05-2002
600288889	Bokkelbakken	A. Christo	P. Polstok	12-12-2002
930563330	Het oude Egypte	P. Halleweg	A. Jannes	12-03-2002
…	…	…	…	…

a. Welke problemen zullen zich voordoen wanneer op deze manier de uitleningen worden geregistreerd?
b. Wat is de oorzaak van deze problemen en hoe wordt dit fenomeen genoemd?
c. De genoemde problemen kunnen worden opgelost door de tabel op te splitsen. In welke tabellen zou de tabel UITLENINGEN moeten worden gesplitst?
d. Maak op papier de tabellen uit vraag c en geef aan met welke sleutels ze naar elkaar verwijzen (voeg waar nodig nieuwe attributen toe om als sleutel te laten functioneren).

Entiteiten, relaties en attributen

- **Opdracht 2**

Een bedrijf wil een database bouwen om haar projecten in bij te houden. De situatie is er als volgt:

> In een bedrijf werken alle werknemers aan projecten. Aan zo'n project werken 1 tot 20 mensen. Iedere werknemer maakt deel uit van maximaal 2 projecten. Men wil graag bijhouden aan welke projecten werknemers werken.

a. Bepaal welke entiteiten (tabellen) er nodig zijn voor de database en teken deze.
b. Bepaal welke relatie(s) (verwijzingen) die entiteiten met elkaar moeten hebben en teken deze tussen de entiteiten van vraag a.

- **Opdracht 3**

> Een school maakt voor het schrijven van lesmateriaal gebruik van docenten. Eén van deze docenten is mevrouw drs. M.Hendriks uit Leiden. Zij heeft tussen 1 oktober 2001 en 6 januari 2003 geschreven aan haar "*Cursus voor beginners*" die de code CVB27 heeft. In de loop der tijd schrijven de meeste docenten heel wat lesmateriaal, maar niet alle docenten schrijven lesmateriaal. Docenten werken altijd in hun eentje aan het lesmateriaal. Natuurlijk wordt niet al het lesmateriaal door docenten geschreven.

a. Welke entiteiten en relaties herken je in deze beschrijving?
b. En welke attributen?
c. Teken het ERD inclusief attributen.

Cardinaliteit

- **Opdracht 4**
  Hieronder vind je een ERD dat is gemaakt voor een theater.

  > Bij een theater wordt bijgehouden welke personen de voorstellingen bezoeken. Een persoon kan in de loop der tijd meerdere voorstellingen bezoeken. Niet iedereen die in het personenbestand is opgenomen heeft al eens een voorstelling bezocht. Gelukkig komt er voor iedere voorstelling altijd publiek opdagen.

  Opdracht: Geef in bovenstaand ERD de functionaliteit aan.

- **Opdracht 5**
  a. Geef in het ERD van opdracht 2 de functionaliteit aan.
  b. Geef in het ERD van opdracht 3 de functionaliteit aan.

- **Opdracht 6**
  a. Geef in het ERD van opdracht 2 de totaliteit aan.
  b. Geef in het ERD van opdracht 3 de totaliteit aan.
  c. Geef in het ERD van opdracht 4 de totaliteit aan.

Complete ER-diagrammen

- **Opdracht 7**

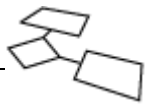
  > In een ziekenhuis liggen alle patiënten op een kamer: soms alleen, soms met meerdere. Het komt wel eens voor dat een kamer leeg is. Van patiënten houdt men een patiëntnummer bij en de naam. Van een kamer wordt een kamercode en het aantal bedden bijgehouden. Tevens wil men vaak weten wanneer een patiënt op de kamer is komen te liggen en wanneer deze er is vertrokken. Patiënten worden wel eens overgeplaatst naar een andere kamer.

  a. Bepaal welke entiteiten, relaties en attributen er nodig zijn om een database voor dit ziekenhuis te bouwen waarin kan worden geregistreerd op welke kamer elke patiënt ligt (of in de loop der tijd heeft gelegen).
  b. Teken het ERD, inclusief cardinaliteit en attributenlijst.

- **Opdracht 8**
  Een bibliotheek wil in een database graag de gegevens over haar boeken gaan bijhouden en geeft je de volgende beschrijving van de organisatie:

  > Van boeken wordt het ISBN en de titel bijgehouden. Ook wordt geregistreerd door welke uitgever het boek is uitgegeven, in welk jaar dit gebeurde en wie het heeft geschreven. Boeken worden wel eens door meerdere auteurs geschreven, maar in deze bibliotheek wordt dan altijd maar één van de auteurs geregistreerd. Van de auteur houdt men alleen de naam bij, van de uitgever wordt de bedrijfsnaam en de vestigingsplaats bijgehouden. Nieuwe uitgevers en auteurs worden pas geregistreerd wanneer ze een boek uitbrengen.

  a. Bepaal welke entiteiten, relaties en attributen er nodig zijn om de database voor deze bibliotheek te bouwen.
  b. Teken het ERD, inclusief cardinaliteit en attributenlijst.



- **Opdracht 9**

  De database die je bij opdracht 8 hebt ontworpen voor de bibliotheek bevalt zo goed, dat men heeft besloten deze door jouw te laten uitbreiden met het uitleensysteem:

  > Van elk boek in de bibliotheek zijn één of meer exemplaren aanwezig (van het boek "*Hocuspocus*" staan bijvoorbeeld maar liefst 8 exemplaren op de plank omdat het erg populair is). Elk exemplaar heeft een uniek exemplaarnummer.
  >
  > Exemplaren kunnen in de loop der tijd meerdere keren worden geleend door de leden van de bibliotheek. Niet ieder exemplaar is al eens uitgeleend. Een lid kan in de loop der tijd natuurlijk meerdere exemplaren lenen, en wordt pas geregistreerd als lid bij de eerste uitlening. Men wil natuurlijk ook graag registreren op welke datum een exemplaar is uitgeleend (daar kan dan uit worden afgeleid wanneer het moet worden teruggebracht).
  >
  > Van leden wordt tot slot nog geregistreerd: het lidnummer (staat ook op de bibliotheekpas), de naam, het telefoonnummer en het adres (straatnaam, huisnummer, postcode en woonplaats).

  Opdracht: Breid het ERD van opdracht 8 nu uit met het hierboven beschreven uitleensysteem.

- **Opdracht 10**

  Na het werk dat je bij opdracht 9 hebt geleverd is men bij de bibliotheek helemaal euforisch. Daarom mag je nu tot slot ook nog het reserveringsysteem ontwerpen:

  > Leden kunnen meerdere boeken reserveren, en een bepaald boek kan zijn gereserveerd door meerdere leden. Van zo'n reservering wordt de datum bijgehouden. Dit is nodig om te bepalen wie voorrang heeft wanneer meerdere leden hetzelfde boek reserveren.

  Opdracht: Breid het ERD van opdracht 9 nu uit met het hierboven beschreven reserveringsysteem.

Bijzondere relaties

- **Opdracht 11**

  Bij de gemeente *Knuppeldam* worden sinds jaar en dag alle panden geregistreerd. Deze registratie verliep altijd op kladblaadjes en is onderhand een redelijke puinhoop geworden. Nu wil men, met de verkiezingen voor de deur, gauw orde op zaken stellen door de panden in een database bij te gaan houden. De burgemeester geeft je de volgende beschrijving:

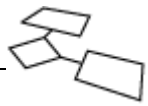
  > Van elke pand wordt het kadasternummer en het adres (straat en huisnummer) geregistreerd. Van bedrijfspanden wordt verder nog de oppervlakte bijgehouden, terwijl men van woonhuizen geïnteresseerd is in het aantal kamers.

  Opdracht: Maak aan de hand van bovenstaande beschrijving een ERD voor de database met panden en red zo de verkiezingscampagne van de partij van de burgemeester.

- **Opdracht 12**

  > Bij een luchtvaartmaatschappij kan een piloot (pilootnummer, naam) begeleid worden door één of meer andere piloten. Een piloot kan nooit meer dan één piloot begeleiden, en niet alle piloten hebben genoeg tijd of ervaring om een andere piloot te begeleiden.

  Opdracht: Maak een ERD voor de database voor pilotenbegeleiding van de luchtvaartmaatschappij.



- **Opdracht 13**

Een groot conferentiecentrum vraagt of je voor hen een conferentiedatabase wil bouwen. Van de organisatie ben je het volgende te weten gekomen:

> Van de conferenties is bekend:
> - datum en plaats
> - op een conferentie spreekt minimaal één spreker, maar meestal spreken er meerdere sprekers
>
> Van de sprekers op de conferenties is bekend:
> - naam en titel (professor, doctor etc.)
> - alle sprekers spreken op één of meer conferenties
>
> Sprekers spreken op zo'n conferentie natuurlijk altijd over een bepaald onderwerp waarvan het volgende bekend is:
> - titel en korte omschrijving van het onderwerp
> - een onderwerp kan in de loop der tijd vaker aan bod komen
> - sommige onderwerpen worden nooit besproken

Opdracht: Maak een ERD voor de conferentiedatabase.

- **Opdracht 14**

Bij de computerchip fabriek *Wintel* staan 643 machines. Men houdt momenteel in Excel-bestanden bij wanneer welke machine wordt gerepareerd, maar deze manier van registreren begint uit de hand te lopen. Daarom wordt jou gevraagd hier op basis van de volgende beschrijving een database voor te ontwerpen:

> Er is een team van zeer gespecialiseerde reparateurs dat reparaties uitvoert op de machines. Een reparateur kan meerdere machines repareren en alle reparateurs hebben al eens een reparatie uitgevoerd. Machines worden in principe altijd door dezelfde reparateur gerepareerd, maar als die afwezig is (ziek, vakantie etc.) door een ander. Van de reparatie wordt de datum geregistreerd, van de reparateur de naam. Sommige machines zijn overigens zo degelijk of nieuw dat ze nog nooit een reparatie hebben hoeven ondergaan.
>
> Van alle machines wordt een machinenummer en de functie van de machine bijgehouden. Van de 43 machines die processors maken wordt tevens bijgehouden welke fabricagetechniek ze gebruiken (0,18 micron, 0,13 micron etc.).

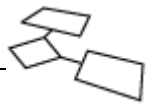
Opdracht: Maak een ERD voor bovenstaande beschrijving.

- **Opdracht 15**

Een informatie-analist heeft onderzoek gedaan naar de abonnementen-registratie van een telefoonmaatschappij. Hieronder volgt een fragment uit het onderzoeksrapport dat hij heeft gemaakt.

> "Bij de telefoonmaatschappij kunnen klanten meerdere abonnementen hebben lopen. Uit een eerste interview bleek dat klanten pas worden geregistreerd als klant wanneer ze een abonnement nemen. Van een klant houdt men de naam en het adres (straat, postcode en woonplaats) bij en de datum waarop elk abonnement is ingegaan. Een bepaald abonnement kan natuurlijk door meerdere klanten worden afgesloten. Sommige abonnementen zijn net nieuw geïntroduceerd en nog door geen enkele klant afgesloten, zo bleek uit de documentatie van de marketingafdeling."
>
> "Er zijn twee soorten abonnementen: voor mobiele telefoons en voor vaste aansluitingen. Van elk abonnement wordt een unieke code, de naam en het maandelijks te betalen bedrag bijgehouden. Verder wordt het simkaart-nummer van mobiele abonnementen, en een voordeelnummer (een door de klant opgegeven telefoonnummer dat tegen lager tarief kan worden gebeld) van vaste aansluitingen bijgehouden. Van beide soorten abonnementen wordt tot slot nog het tarief (per seconde) bijgehouden."

Opdracht: Maak een ERD van de abonnementregistratie van deze telefoon-maatschappij.



- **Opdracht 16**

Woningbouwvereniging *Sloopzicht* wil haar gegevens over leden en hun inschrijvingen in een database gaan bijhouden. De directeur geeft je de volgende informatie:

> Momenteel worden de gegevens van leden op ledenkaarten geregistreerd. Wanneer een lid zich inschrijft wordt, naast de ledenkaart, ook een zogenaamd inschrijfformulier ingevuld (zie onderstaande afbeeldingen).

Woningbouwvereniging *Sloopzicht*

LEDENKAART

Lidnummer	:	5689	**Geboortedatum**	:	12-06-1980
Naam	:	A. De Bruin	**Telefoon**	:	(0233) 564954

Woningbouwvereniging *Sloopzicht*

INSCHRIJFFORMULIER

Datum : 02-02-2002

Lidnummer : 5689
Naam : A. De Bruin
Geboortedatum : 12-06-1980
Telefoon : (0233) 564954

Schrijft zich in als woningzoekende voor:

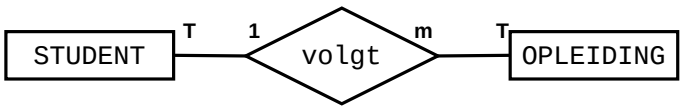
Type woning : Eengezinswoning
Stadsdeel : Oost-Lommeren
Max. huurprijs : € 450,-
Opmerkingen : Graag inclusief zwembad

a. Maak op basis van bovenstaande informatie en formulieren een ERD voor de database van de woningbouwvereniging.
b. Stel nu dat een lid zich *tegelijk* kan inschrijven voor *meerdere* type woningen en *meerdere* stadsdelen. Maak een ERD voor deze nieuwe situatie.

Vertalen naar relationeel model

- **Opdracht 17**

Een school wil een database bouwen voor het bijhouden van haar student- en opleidinggegevens. Hieronder vind je het ontwerp dat daarvoor is gemaakt:



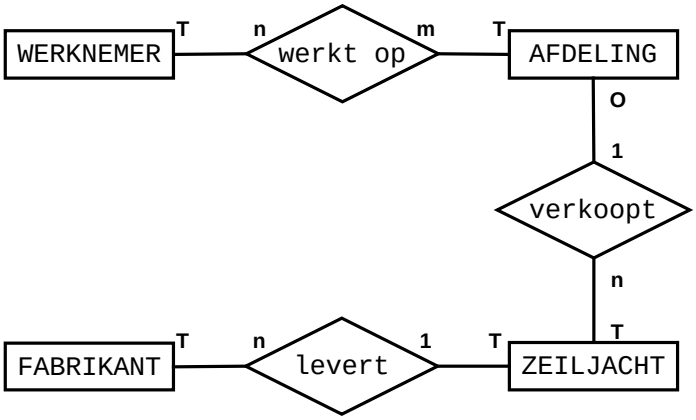
Attributenlijst

```
STUDENT     : studentnummer, naam, telefoon
OPLEIDING   : opleidingscode, naam
volgt       : cohort
```

a. Pas op bovenstaand ERD de drie basisregels toe.
b. Herstructureer vervolgens het relationeel model, zodat het ontwerp helemaal klaar is en de database gebouwd kan worden.

- **Opdracht 18**

Het ERD hieronder is gemaakt voor een bedrijf dat zeiljachten verkoopt. Er zijn een aantal verkoopafdelingen die elk gespecialiseerd zijn in het verkopen van één bepaald type zeiljacht.

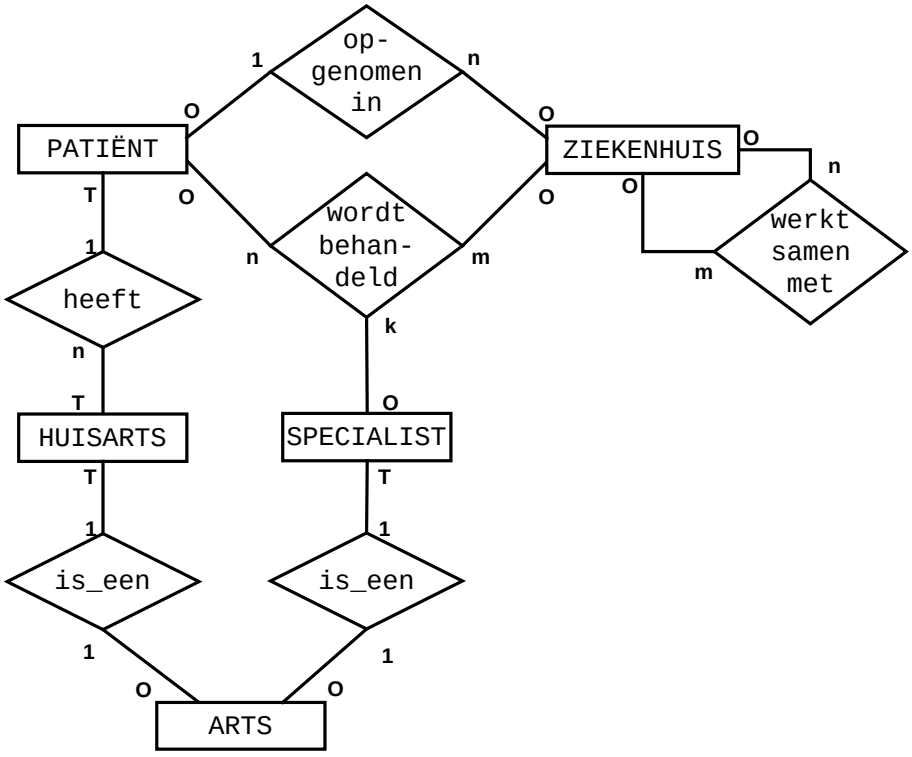


Attributenlijst

```
WERKNEMER       : naam
AFDELING        : afdelingscode, naam, telefoon
ZEILJACHT       : omschrijving, adviesprijs
FABRIKANT       : naam, e-mail
verkoopt        : verkoopprijs
levert          : inkoopprijs
```

a. Pas op bovenstaand ERD de basisregels toe.
b. Herstructureer vervolgens het relationeel model.

- **Opdracht 19**



Attributenlijst

```
PATIËNT         : pcode, naam
ZIEKENHUIS      : zcode, naam, plaats
ARTS            : acode, naam, woonplaats
HUISARTS        : praktijkgrootte
SPECIALIST      : specialisme
opgenomen in    : datum
```

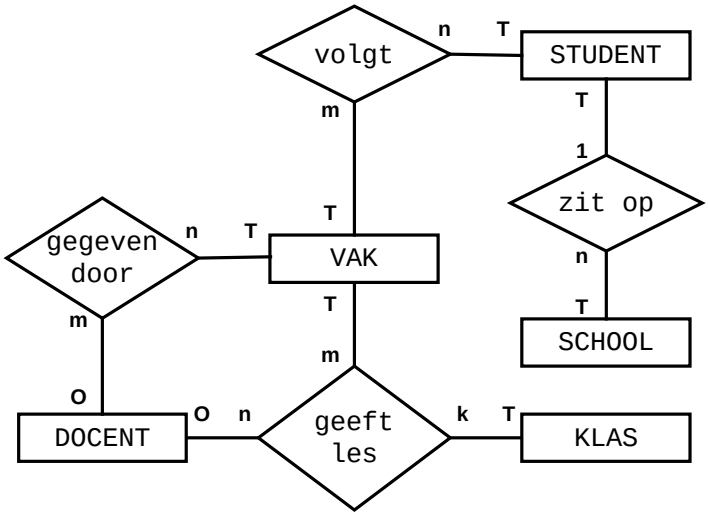
Opdracht: Vertaal bovenstaand ERD naar het relationele model.

ER-diagrammen ontwerpen

- **Opdracht 20**

  Onderstaand ERD is gemaakt voor een school op basis van de volgende beschrijving:

  > De school wil graag bijhouden welke studenten er studeren en in welke klas iedere student zit. Verder wil men weten welke klas welk vak volgt bij welke docent.

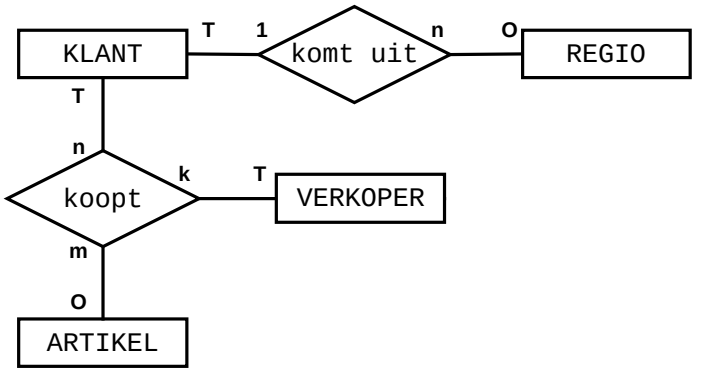


  Opdracht: Verbeter bovenstaand ontwerp zodanig dat het aansluit bij de beschrijving en een goede database zal opleveren.

- **Opdracht 21**

  Onderstaand ERD is gemaakt voor de marketingafdeling van een groothandel op basis van de volgende beschrijving:

  > De marketingafdeling is geïnteresseerd in een aantal gegevens van klanten (hun naam en de regio waar ze vandaan komen en in welk land die regio ligt) en wanneer zij bepaalde artikelen hebben gekocht. Van artikelen houdt men de prijs en omschrijving bij en in welke artikelgroep zo'n artikel zit.



Attributenlijst

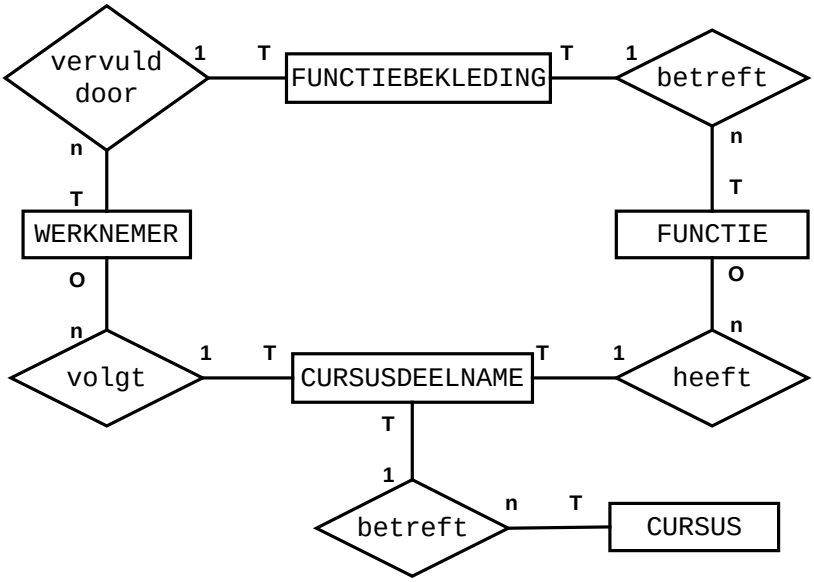
```
KLANT     : naam, land
REGIO     : naam
VERKOPER  : naam
ARTIKEL   : omschrijving, prijs, artikelgroep
koopt     : datum
```

Opdracht: Verbeter bovenstaand ontwerp zodanig dat het aansluit bij de beschrijving en een goede database zal opleveren.

- **Opdracht 22**

Onderstaand ERD is gemaakt voor de volgende situatie:

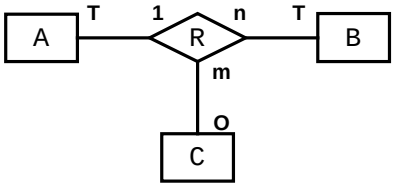
> Men wil van werknemers registreren welke functies ze bekleden en aan welke cursussen ze in dat kader hebben deelgenomen. Men is dus geïnteresseerd in de volgende relaties:
> - welke functies een werknemer in de loop der tijd allemaal heeft bekleed
> - aan welke cursussen een werknemer heeft deelgenomen toen hij een bepaalde functie bekleedde



Opdracht: Bovenstaand ERD voldoet op zich maar is onnodig ingewikkeld. Maak het ontwerp eenvoudiger.

- **Opdracht 23**

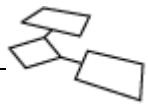
Stel dat we aan de hand van onderstaand ERD besluiten een database te bouwen met de tabellen A, B, C en R.



Attributenlijst na herstructureren

A: As
B: Bs
C: Cs
R: As, Bs, Cs

Opdracht: Als de tabel R drie records bevat, hoeveel records bevat elk van de overige tabellen dan maximaal en hoeveel minimaal?



Extra opdrachten

- **Opdracht 24**

  Streekvervoerbedrijf *Claxxion n.v.* heeft je gevraagd een database te ontwerpen voor de volgende situatie:

  > Het bedrijf beschikt over autobussen en een klein aantal taxi's. Het gebied waarin het bedrijf opereert is opgedeeld in een aantal rayons. Elke autobus en elke taxi wordt onderhouden door één rayon. Elk rayon onderhoudt een aantal autobussen en taxi's. Men wil de volgende gegevens bijhouden:
  > - van een autobus: voertuigcode, kenteken, omschrijving, merk, type en de personencapaciteit
  > - van de rayons: rayoncode en omschrijving
  > - van de taxi's: kenteken, voertuigcode, merk, type, omschrijving en de brandstofsoort (gas, benzine etc.)
  >
  > In zo'n rayon werkt een aantal medewerkers, waarbij alle medewerkers (behalve de directeur natuurlijk) rapporteren aan de medewerker die boven ze staat. De medewerkers zijn in elk rayon inzetbaar. Van elke medewerker wil men naam, adres, postcode en woonplaats registreren.

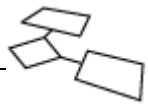
  Opdracht: Maak aan de hand van bovenstaande beschrijving een ERD voor *Claxxion n.v.* en vertaal het naar het relationele model.

- **Opdracht 25**

  Groothandel *MultiProcz* handelt in computeronderdelen. Ze verkopen processors, cd-rom spelers, monitoren etc. aan computerwinkels in de regio. Ze zijn toe aan een nieuw verkoopsysteem en geven daarvan de volgende beschrijving:

  > Wanneer een medewerker van de afdeling verkoop 's ochtends op zijn werk komt drinkt hij eerst een kop koffie bij de koffieautomaat en neemt vervolgens plaats achter zijn bureau. Wanneer een klant belt legt hij de artikelen die de klant noemt vast in een verkooporder. Zo'n verkooporder bestaat uit een aantal orderregels, waarbij in elke orderregel het aantal bestelde artikelen en de prijs per stuk wordt opgenomen. Van de verkooporder wordt verder nog vastgelegd: een uniek ordernr en de datum waarop hij is geplaatst. Met de klant wordt een afspraak gemaakt wanneer de order uiterlijk moet worden geleverd.
  >
  > Van medewerkers en klanten houdt men de naam, het adres, de postcode en de woonplaats bij. Van medewerkers wordt ook nog het salaris en de medewerkercode geregistreerd, van klanten de kredietlimiet en een uniek klantnr. Het totaalbedrag dat klanten aan bestellingen open hebben staan mag niet uitstijgen boven hun kredietlimiet. Nieuwe klanten worden pas in het klantenbestand opgenomen wanneer ze hun eerste order plaatsen.
  >
  > De artikelen zijn voor het gemak ingedeeld in artikelgroepen, bijvoorbeeld een artikelgroep "processors" en een artikelgroep "vaste schijven." Van artikelen wordt de prijs, de omschrijving en een unieke artikelcode bijgehouden. Ook wil men natuurlijk graag weten hoeveel stuks er van elk artikel in voorraad zijn en is er vaak behoefte om van een artikel extra gegevens vast te leggen (bijvoorbeeld van een bepaalde processor dat deze niet werkt met Windows 2000, of van een monitor dat deze blauwe stippen op de zijkant heeft etc.).

  Opdracht: Maak aan de hand van bovenstaande beschrijving een ERD voor *MultiProcz* en vertaal het naar het relationele model.



- **Opdracht 26**

  Op *Schiphol* is men toe aan een nieuw terminalsysteem: het systeem dat verantwoordelijk is voor het vastleggen van de vluchtgegevens en deze via monitoren toont aan de reizigers. Het betreft hier een actuele database, het gaat er immers alleen om de huidige vluchtgegevens op de monitoren weer te geven, wat er in het verleden is gebeurd is (voor dit systeem) niet interessant. Het systeem zit als volgt in elkaar:

  Om de juiste informatie op de monitoren te kunnen weergeven moet men van elke vlucht weten welke maatschappij de vlucht uitvoert met welk type toestel. Van elke vlucht wordt het vluchtnummer bijgehouden. Tevens wordt de vertrektijd en de huidige status (die gecodeerd wordt opgeslagen: D=delayed, B=boarding, T=taking off etc.) van de vlucht geregistreerd.

  Ook de luchthaven (luchthavencode, plaatsnaam) waar de vlucht vandaan komt en de luchthaven waar naartoe wordt gevlogen worden opgeslagen. Niet iedere luchthaven wordt als herkomst of bestemming van een vlucht gebruikt.

  Van elk toestel is verder bekend wat het (unieke) type (B737, A320 etc.) en het aantal zitplaatsen is. Van elke maatschappij wordt alleen het maatschappijnr en de naam (Iberia, KLM etc.) bijgehouden. Niet iedere maatschappij (en niet elk toestel) in het systeem voert vluchten via *Schiphol* uit.

  De terminalschermen op de luchthaven moeten er uit komen te zien zoals hieronder is weergegeven:

Departures — time 12:42

flight		departure	from	to	plane	status
IB3485	Iberia	12:30	Amsterdam	Madrid	A320	
BA6654	British Airways	12:40	Singapore	London	B737	*delayed*
KL644	KLM	12:45	Amsterdam	Frankfurt	B737	*boarding*
IB6556	Iberia	13:05	Amsterdam	Lima	A340	
TR1324	Transavia	13:10	Frankfurt	New York	B747	

Opdracht: Ontwerp een ERD voor de actuele database van het terminal-systeem. Vertaal jouw ERD tevens naar het relationele model zodat de database (en vervolgens de terminalhardware en -software) kan worden gebouwd.