Cursus Databasedesign

Inhoudsopgave

Hoofdstuk 1 Database ontwerpen 2
Hoofdstuk 2 Relaties 8
Hoofdstuk 3 De weg naar een goede ERD 15
Hoofdstuk 4 Geschiedenis en supertype subtype 23
Hoofdstuk 5 Hiërarchische en Recursieve relaties 27

Hoofdstuk 1 Database ontwerpen

In deze cursus leer je hoe je een informatiesysteem kunt ontwerpen en bouwen waarmee je gegevens kunt opslaan en beheren op een dusdanige manier dat je er snel en zonder fouten informatie uit kunt halen.

We richten ons hierbij vooral op het ontwerpen van een relationele database. Merk op dat er bij NoSQL databases meer ontwerpvrijheden zijn dan hieronder beschreven.

Termen

Consulting

Informatie analist

Proces analist

Logisch ontwerp

Fysiek ontwerp

Entiteit

Attribuut

Waarde

Null

Datatype

Vluchtig (Engels: volatile)

Verplicht (Engels: mandatory)

Optioneel

primary UID (unieke identifier)

secondary UID

PK (primary key)

Inleiding

Bij het ontwikkelen van een informatiesysteem kan men vier fasen onderscheiden:

1. Analyse
2. Logisch ontwerp
3. Fysiek ontwerp
4. Bouw

Analyse

Consulting
Om een informatiesysteem te kunnen ontwerpen moet je gesprekken voeren met de klant waarin deze zijn/haar wensen kenbaar kan maken. Dat is lastiger dan het lijkt want de klant heeft er vaak geen idee van hoe een database werkt. Twee dingen zijn hierbij belangrijk:

- Stel de juiste vragen aan de klant.
- Stel ook de vragen waar de klant niet aan denkt.

Het beroep wat hier bij hoort heet consultant. Zij moeten de vertaalslag maken van wat de klant wil, naar degenen die het uiteindelijk moeten gaan maken. Mensen die dit goed kunnen zijn schaars en worden zeer goed betaald.

Bij de analyse zijn twee soorten consultanten nodig:

- De informatie-analist: Deze houdt zich vooral bezig met het ontwerpen van de database.
- De proces-analist Deze houdt zich vooral bezig houdt met welke processen er in het bedrijf spelen en wat er op de verschillende gebruikersschermen zichtbaar moet worden.

Bij kleine projecten is er één consultant die beide doet.

Logisch ontwerp

Vanaf dit punt gaan we uit van het ontwerp van een relationele database.

Een logisch model:
- modelleert functionele en informatieve behoeftes
- is gebaseerd op huidige behoeftes en houdt alvast rekening met toekomstige behoeftes
- gaat enkel over business needs, heeft dus niets van doen met de implementatie
- noemen we ook wel een Entity relationship Model (ERM), zie hoofdstuk 2.
- wordt getoond met een Entity Relationship Diagram (ERD), zie hoofdstuk 2.
- nodigt uit tot discussie
- voorkomt fouten en misvattingen
- vormt meteen een ideaal documentatiesysteem
- geeft een goede basis voor een fysiek database design
- documenteert de business rules (de processen van een organisatie)
- houdt rekening met regels en wetten
- is niet gericht op een bepaald soort techniek (DBMS, platvorm)
- gebruikt voor het bedrijf begrijpelijke benamingen

Fysiek ontwerp

Op basis van het logisch ontwerp wordt het fysiek ontwerp gebouwd. In tegenstelling tot het logische model is het fysieke model wel gericht op één van de belangrijkste DBMS-typen relationeel, object-georiënteerd en NoSQL.

Een fysiek model:

- richt zich op een bepaald type DBMS
- bevat bij een relationele database de structuur van de tabellen
- bevat alle kenmerken van de gegevens
- richt zich ook op de details van de implementatie

Entiteiten, attributen en waarden

Een logisch ontwerp bestaat uit entiteiten en attributen. In een fysiek ontwerp komt dit overeen met de tabellen en de kolomnamen.

Een entiteit is het object waarvan informatie wordt opgeslagen.

Een attribuut is een eigenschap.

Logisch ontwerp

Uiteindelijke tabel

Leerlingen			
voornaam	achternaam	geboortedatum	telefoonnr
Piet	Konijn	13-4-2012	0611211211
Jan	Haas	15-2-2012	
Sylvia	Slak	23-7-2011	0600700700

In het voorbeeld hierboven is leerling een entiteit en geboortedatum een attribuut. Wanneer je de database gaat bouwen, vul je de rijen van de tabellen met waarden (Engels: values). Konijn is dus een waarde.

Een leeg vakje vakje betekent dat er geen waarde is. Je noemt de waarde dan null.

Null betekent dus geen waarde. Dat is wat anders als het getal 0 want dat is wel een waarde.

Wanneer je een entiteit tekent moet je aan bepaalde regels voldoen:

1. Gebruik rechthoeken met afgeronde hoeken.
2. De entiteitnaam moet in hoofdletters en enkelvoud
3. De attribuutnamen moet in kleine letters

Fysieke ontwerp van tabel leerlingen			
Kolomnaam	**Datatype**	**Lengte**	**Standaardwaarde**
v_naam	varchar	20	
a_naam	varchar	30	
geboortedatum	date		
tel_nr	char	10	

Hierboven zie een voorbeeld van een fysiek model van de tabel leerlingen. Er zijn meer kolommen maar deze volgen verderop in het verhaal.

De kolomnamen zijn wat afgekort om de tabellen overzichtelijk te houden. In het fysieke model wordt aan iedere kolomnaam ook een datatype (format) toegevoegd.

Een datatype is het soort data dat wordt opgeslagen. Dit kan zijn een getal (integer, kommagetal), string (varchar, char), datum (date), plaatje, geluid enzovoort.

Met de lengte bedoelen we bij een string uit hoeveel tekens deze maximaal mag bestaan.

Met de standaardwaarde bedoelen we dat je standaard al een waarde hebt ingevuld. Wanneer je een kolomnaam “nationaliteit” hebt en bijna alle klanten hebben de Nederlandse nationaliteit dan zou je hier “NL” kunnen invullen. Slechts enkele klanten hoeven dit veld dan aan te passen in hun formulier.

Vluchtige attributen

Niet alle eigenschappen zijn even geschikt om te gebruiken in een database. In bovenstaand voorbeeld zou je bijvoorbeeld de eigenschap “leeftijd” kunnen gebruiken in plaats van geboortedatum. Dat is niet echter handig want dan moet je dit na iedere verjaardag bijwerken. Zulke eigenschappen heten “vluchtige” eigenschappen.

Verplicht of optioneel

Sommige attributen moeten altijd worden ingevuld. In het voorbeeld hierboven is dat bijvoorbeeld “achternaam”. Dit zijn verplichte attributen. Andere zijn optioneel en dus niet verplicht. In het voorbeeld hierboven is dat bijvoorbeeld telefoonnummer. Niet iedereen heeft een telefoonnummer of dit nummer is geen noodzakelijk gegeven.

UID Unieke identifier

In een database wil je perse dat iedere rij uniek is. Geen enkele regel mag exact hetzelfde zijn. Om te garanderen dat dit het geval is maakt men gebruik van unieke identifiers. Dit kan een attribuut zijn of combinatie van attributen en zelfs een relatie kan deel uitmaken van een UID. In het voorbeeld hierboven kun je denken aan “achternaam”. Op een school zitten echter verschillende leerlingen met dezelfde achternaam. Een betere UID zou zijn de combinatie van voor en achternaam maar het komt regelmatig voor dat leerlingen dezelfde voor en achternaam dragen. Een goede UID zou zijn de combinatie van voornaam, achternaam en geboortedatum. De kans dat twee verschillende leerlingen dit hetzelfde hebben is zeer klein. Om alle risico’s uit te sluiten voegt men echter vaak een attribuut toe. In dit geval “leerlingnummer”. Door ieder jaar andere nummers te gebruiken is persoonsverwisseling uitgesloten.

In het fysieke model heet de UID de primary key (PK).

De entiteit LEERLING zou er als volgt uit kunnen zien:

= primary UID, dit is de UID die uiteindelijk gebruikt gaat worden.

(#) = secondary UID, deze UID heeft men als reserve en wordt gebruikt als controlemiddel. Hier is dat een combinatie van voornaam, achternaam en geboortedatum. Een secondary UID past men vaak toe als de primary UID geen herkenbare naam heeft die in verband staat met de naam van de entiteit.

* = verplicht

o = optioneel

Opgaven

Opgave 1

Zet de volgende vier fasen van het ontwikkelen van een informatiesysteem op volgorde van begin naar eind:

A Fysiek model B Bouw C Analyse D Logisch model

Opgave 2

Waar of niet waar: Een logisch model houdt rekening met het type DBMS waarmee de database gebouwd gaat worden.

Opgave 3

Welke zijn de drie belangrijkste DBMS-typen?

Opgave 4

Voetbalclub	**Stad**	**Aantal gespeelde wedstrijden**	**Aantal punten**
FC Emmen	Emmen	3	0

Waar of niet waar: De waarde van het aantal punten in de rij hierboven is null.

Opgave 5

Hieronder staat een verhaaltje. Met dit verhaaltje wil je een database ontwerpen. Geef aan welke woorden uit dit verhaal je als entiteit zou gebruiken, welke als eigenschap en welke als waarde? Noem ook de eigenschappen die niet in de tekst genoemd staan maar waarvan wel waarden zijn genoemd.

Bij autohandel “Krakkemik” staan er auto's van verschillende merken op het terrein. Zo staat er een rode Opel Astra uit 1998 voor 1500 euro, een groene Ford Escort uit 2002 voor 2000 euro en een Peugeot waarvan de prijs 4000 euro is.

Opgave 6
Teken de entiteit die bij de opgave uit de vorige vraag hoort. Bedenk zelf de attributen die niet in de tekst voorkomen. Geef aan welke attributen tot de UID behoren, welke verplicht zijn en welke optioneel.

Opgave 7
Geef bij ieder attribuut uit de vorige opgave aan welk datatype en lengte je zou gebruiken in het fysieke model. Op https://www.techonthenet.com/mariadb/datatypes.php vind je een lijst met datatypes waaruit je kunt kiezen. Als deze pagina niet meer werkt zoek je op “datatypes mysql”.

Opgave 8
Leg uit waarom in iemands paspoort niet diens haarlengte staat opgegeven.

Hoofdstuk 2 Relaties

Termen

Relatie

Kardinaliteit

Erdish

Kraaienpootnotatie

ERD

ERM

Constraint

Overdraagbaarheid (transferability)

Niet aanpasbaar (not updatable)

Bedrijfsapplicatie

Module

CRUD analyse

Relaties

Wanneer er sprake is van twee of meer entiteiten dan kunnen deze entiteiten een relatie hebben.

Relaties:

- tonen iets belangrijks
- geven aan hoe entiteiten zich tot elkaar verhouden
- komen in paren; van A naar B en van B naar A
- hebben een mate van kardinaliteit

Kardinaliteit wil zeggen: hoeveel?

Erdish

De relaties schrijven we op een bepaalde manier op zodat we ze later makkelijk in het ERD kunnen tekenen.

Voorbeelden:

Iedere werknemer heeft één of meer banen.

Iedere baan wordt door precies één werknemer vervuld.

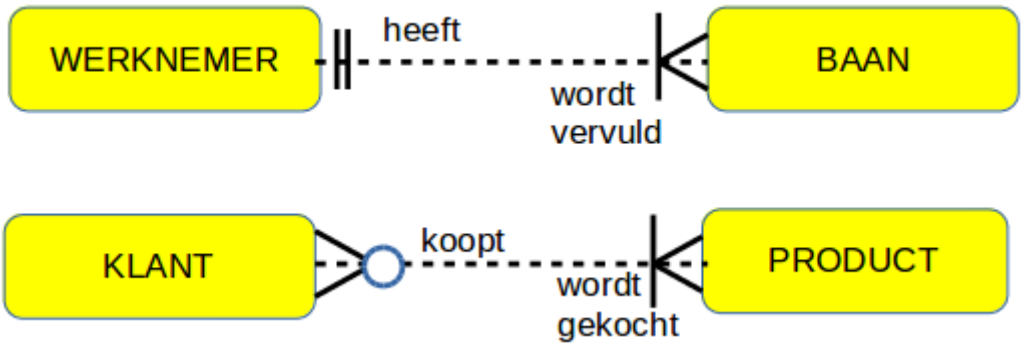
Iedere klant koopt één of meer producten.

Ieder product wordt gekocht door nul of meer klanten.

Toelichting:

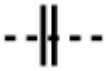
- Iedere zin begint met “iedere” of “elke”.
- De zinnen komen in paren.
- Er zit een werkwoord in de zin.
- Eén of meer, precies één en nul of meer geeft de kardinaliteit aan.
- Vaak moet met de opdrachtgever besproken worden hoe het precies zit. Is het bijvoorbeeld toegestaan dat een werknemer meerdere banen heeft.

Tekenafspraken



Er bestaan meerdere notaties maar wij gebruiken de kraaienpoot notatie omdat die het meest intuïtief is.

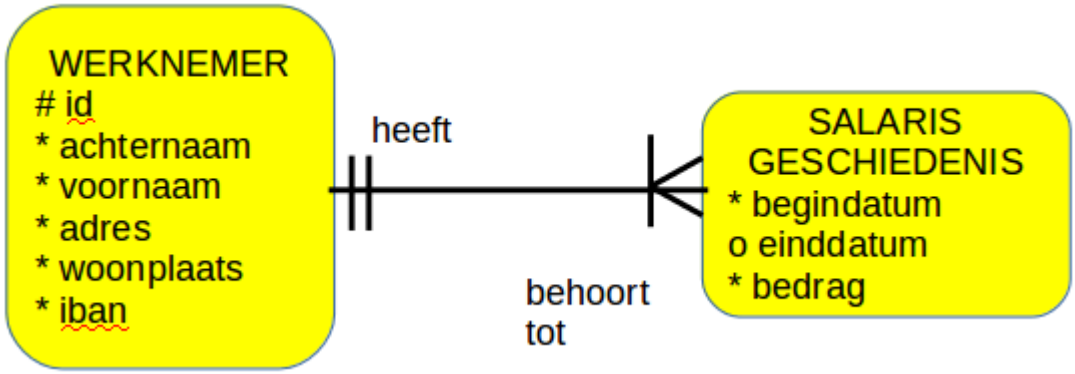
 Nul of één

 Precies één

Nul of meer


Eén of meer

Hierbij erft de kind entiteit de UID van de ouder entiteit; hieronder staat een voorbeeld. Een nadeel van deze notatie is dat je niet in één oogopslag kunt zien wat de ouder entiteit is en wat de kind entiteit.



- Hierboven geeft de doorgetrokken lijn aan dat de UID van de ouder entiteit WERKNEMER wordt overgedragen op de kind entiteit SALARISGESCHIEDENIS.
- De UID van SALARISGESCHIEDENIS is dus werknemer_id.
- Merk op dat de UID van SALARISGESCHIEDENIS hier dus uit een relatie bestaat.
- In het fysieke model komt hiervoor een extra kolom werknemer_id in de tabel SALARISGESCHIEDENIS.

ERD

Wat je hierboven getekend ziet is een eenvoudig ERD, een Entity Relatitonship Diagram. Een ERD geeft de entiteiten weer die van belang zijn en de relaties die tussen deze entiteiten bestaan.

- Het doel van een ERD is om een voorstel te documenteren waarover discussie kan plaatsvinden.
- Informatie mag slechts één keer worden getoond in een ERD,

- Informatie die van andere informatie kan worden afgeleid moet je niet in het model stoppen. Wanneer bijvoorbeeld geboortedatum genoemd wordt hoef je niet nog eens leeftijd te noemen want die kan uit geboortedatum berekend worden.

ERM

Een ERM Entity Relationship Model is een logisch model voor relationele databases.

- Een ERM bevat meestal een ERD maar niet altijd.
- Bevat ook informatie die niet in de ERD opgenomen kunnen worden zoals datatypen en bepaalde constraints (beperkingen).
- Is onafhankelijk van de implementatie van hardware en software.

Constraints

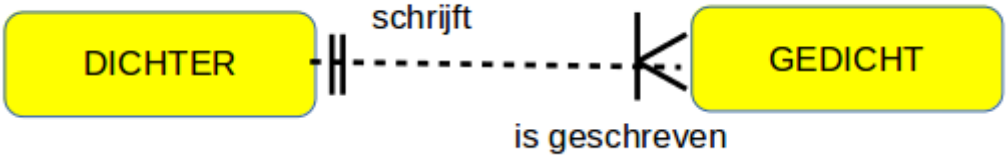
Constraints zijn beperkingen of voorwaarden die voortvloeien uit de bedrijfsregels. Stel dat een bedrijf wil dat het informatiesysteem een seintje geeft als een werknemer 25 jaar bij het bedrijf werkt, zodat er iets georganiseerd kan worden, dan moet dit na de bouw van de database erbij worden geprogrammeerd. Dit staat dus niet in de ERD maar wordt wel gedocumenteerd in het ERM.

Andere voorbeelden zijn: Er mogen geen geboortedata worden ingevoerd die in de toekomst liggen. De begindatum moet voor de einddatum liggen. Een afdeling moet uit minimaal drie werknemers bestaan. Dit kun je voor elkaar krijgen door wat extra programmeerwerk.

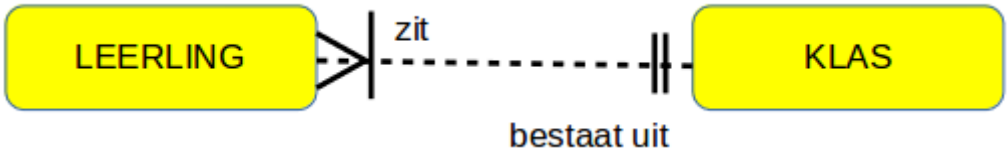
Andere constraints zoals welke attributen tot de UID behoren kun je wel in de ERD opnemen.

Overdraagbaarheid

Sommige relaties zijn niet overdraagbaar (transferable). Het komt alleen voor aan de “meer” kant van één meer relaties. Dit moet in de het ERM gedocumenteerd worden. In het fysieke model maak je vervolgens de relatie aan die kant niet aanpasbaar (not updatable). Bekijk het voorbeeld hieronder.



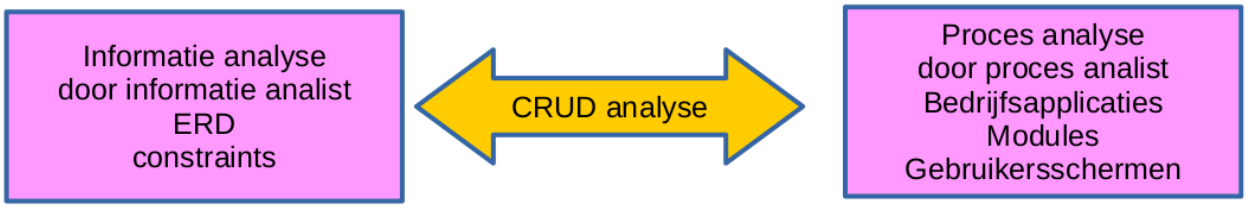
Een gedicht dat eenmaal door een bepaalde dichter is geschreven kan later niet door een andere dichter zijn geschreven. Er moeten maatregelen genomen worden zodat dit niet aangepast kan worden. De relatie van gedicht naar dichter is dus niet overdraagbaar. De relatie van dichter naar gedicht is wel overdraagbaar omdat de dichter meerdere gedichten heeft geschreven en hij dus het ene gedicht kan uitwisselen met het andere.



De relaties hierboven zijn beiden overdraagbaar want een leerling kan achteraf naar een andere klas worden overgeplaatst.

CRUD analyse

Een ERD zal uiteindelijk leiden tot een verzameling tabellen waarin we gegevens als rijen gaan opslaan (de database). Een database vormt de basis van een bedrijfsapplicatie. Een bedrijfsapplicatie bestaat uit verschillende modules. Bijvoorbeeld een module “Klant gegevens onderhouden”.



Een ander onderdeel van het ERM is de CRUD analyse per module van de bedrijfsapplicatie. Genoteerd moet worden welke data in die module gemaakt, opgehaald, aangepast en verwijderd moet kunnen worden.

Create	Welke data wordt aangemaakt?
Retrieve	Welke data wordt opgehaald?
Update	Welke data wordt ververst?
Delete	Welke data wordt gewist?

Opgaven

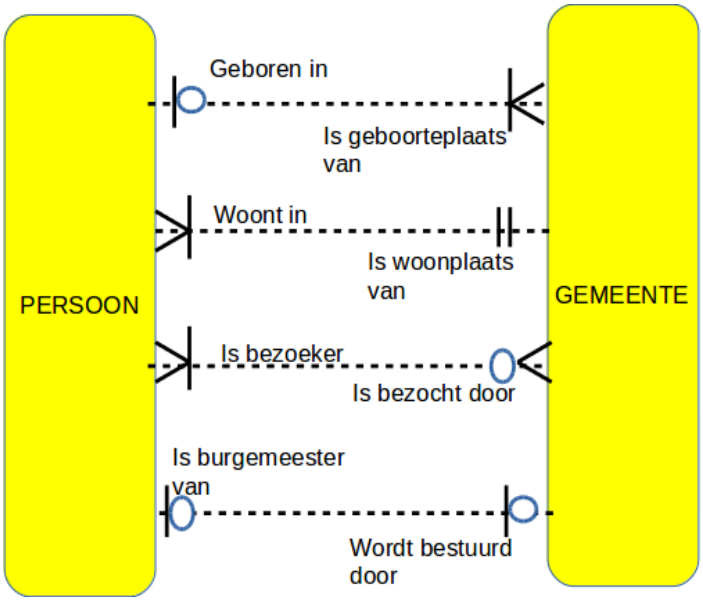
Opgave 1 Erdish zinnen maken

Maak steeds twee Erdish zinnen over de relaties tussen de twee entiteiten.

a) Leerling Klas
b) Leerling Stoel
c) Fototoestel Foto
d) Gedicht Dichter

Opgave 2 ERD lezen

a) Schrijf de Erdish zinnen op bij onderstaande relaties in het ERD.
b) Er staan een paar relaties tussen die in werkelijkheid niet kloppen. Schrijf achter iedere zin of de relatie kan of niet kan is.



Opgave 3 Tekenafspraken

Teken ERD's bij de volgende relaties.

a) Iedere woonplaats is de geboorteplaats van nul of meer personen. Iedere persoon is geboren in precies één woonplaats.
b) Iedere kamer herbergt nul of meer gasten. Iedere gast logeert in precies één kamer.
c) Iedere werknemer werkt op precies één afdeling. Iedere afdeling heeft één of meer werknemers.
d) Iedere e-mail is gericht aan één of meer personen. Ieder persoon is de geadresseerde van nul of meer e-mails.
e) Ieder stuk gereedschap heeft precies één prijs. Iedere prijs hoort bij één of meer stukken gereedschap.
f) Ieder kind heeft precies één moeder. Iedere moeder heeft één of meer kinderen.
g) Iedere leerling heeft les van één of meer docenten. Iedere docent geeft les aan één of meer kinderen.
h) Iedere vingerafdruk behoort tot precies één persoon. Ieder persoon heeft één of meer vingerafdrukken.

Opgave 4 Overdraagbaarheid

Geef aan welke van de relaties uit de vorige opgave niet overdraagbaar zijn.

Opgave 5 Crud analyse: Create, Retrieve, Update, Delete

In computerprogrammaś komen allerlei handelingen voor die allemaal te herleiden zijn tot één van de vier CRUD acties. Zet het juiste CRUD letter achter iedere term.

a) Alter
b) Bring up
c) Change
d) Discard
e) Enter
f) Find
g) Import
h) Input
i) Load
j) Look up
k) Modify
l) Print
m) Purge
n) Read
o) rij
p) Remove
q) Report
r) Trash
s) View

Hoofdstuk 3 De weg naar een goede ERD

Termen

Matrix

Normalisatie

Eerste normaalvorm

Tweede normaalvorm

Derde normaalvorm

Meer meer relaties (Engels: many to many relationship)

Intersection entiteit

Matrix

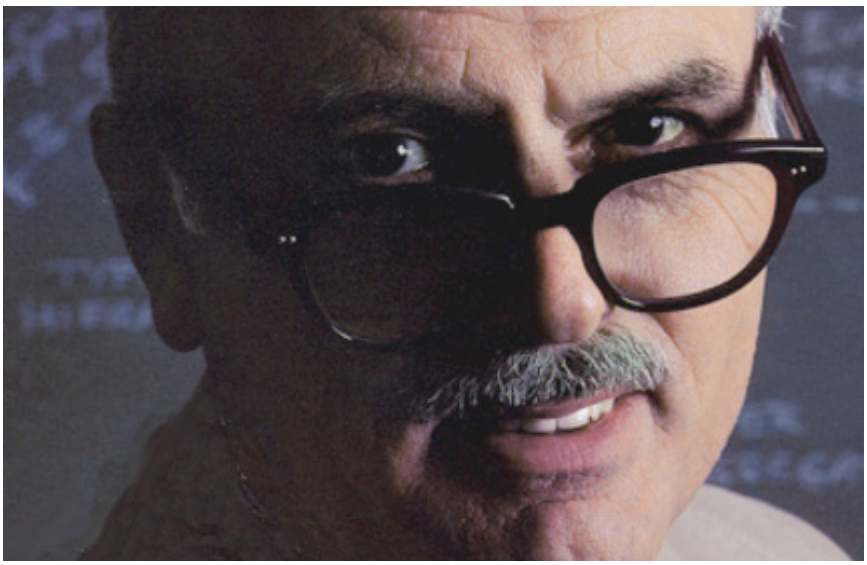
Het komt vaak voor dat een ERD uit meer dan twee entiteiten zal bestaan. In de analyse komen er allerlei kandidaat entiteiten naar voren. Alleen entiteiten waartussen een relatie bestaat worden in de ERD opgenomen. Een goed hulpmiddel hierbij is een matrix. Hieronder staat een voorbeeld.

	REIZIGER	LAND	BEZIENSWAARDIGHEID	BETAALMIDDEL
REIZIGER	x	bezoekt	wordt bezocht	betaalt met
LAND	wordt bezocht	x	bezit	x
BEZIENSWAARDIGHEID	wordt bezocht	ligt in	x	wordt betaald met
BETAALMIDDEL	wordt gebruikt	x	wordt gebruikt	x

Een matrix helpt bij het vinden van alle mogelijke relaties tussen entiteiten.

Normalisatie

Bij het bepalen van de juiste entiteiten en attributen kan normaliseren een belangrijk hulpmiddel zijn. Hiervoor zijn door de Amerikaan Ted Codd regels opgesteld waarvan hier de drie belangrijkste worden toegelicht. Stapsgewijs helpen ze om belangrijke fouten in het ontwerp te ontdekken.

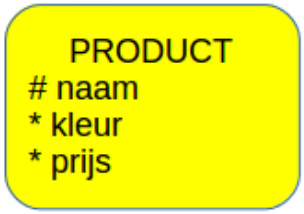


Ted Codd

Eerste normaalvorm

Een entiteit is in de eerste normaalvorm als de attributen niet meer dan één waarde kunnen hebben per exemplaar van de entiteit en de attributen zelf ook maar één keer voorkomen.

Voorbeeld



Hier lijkt niets mee aan de hand te zijn want de opdrachtgever heeft geen producten met dezelfde naam. Het probleem komt aan het licht als de informatie-analist de fabrikant een proeftabel laat invullen.

PRODUCT

naam	kleur	prijs
badeendje	rood of groen of blauw	7,95
octopus	oranje of blauw	13,55
kikker	groen	19,95

Blijkbaar kan een product met dezelfde naam meerdere kleuren hebben. Per exemplaar van de entiteit zijn er dus meerdere waarden Dit is als volgt op te lossen:

Er is een unieke eigenschap “id” toegevoegd. De secondary UID zorgt ervoor dat een product nog steeds op een combinatie van naam en kleur kan worden opgezocht wat handig kan zijn als een klant het product_id niet weet.

Hieronder staat de bijbehorende proeftabel:

PRODUCT

id	naam	kleur	prijs
1	badeendje	rood	7,95
2	badeendje	geel	7,95
3	badeendje	blauw	7,95
4	octopus	oranje	13,55
5	octopus	blauw	13,55
6	kikker	groen	19,95

In de proeftabel zie je dat ieder attribuut per exemplaar van de entiteit slechts één waarde kan hebben. Een voordeel voor de fabrikant is verder dat hij een populaire kleur badeendje een hogere prijs kan geven.

Bij de badeendjes hoef je niet ieder afzonderlijk exemplaar in je database te stoppen. Bij auto’s is dat anders want iedere auto heeft een uniek chassisnummer. Overleg met de opdrachtgever is dus belangrijk om tot een goede database te komen.

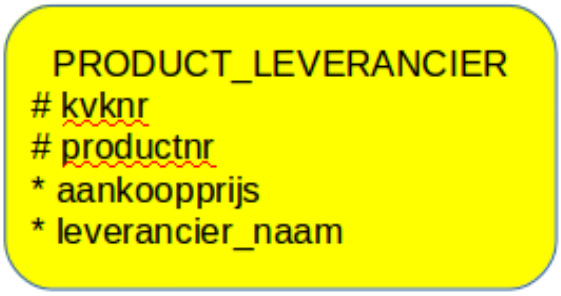
Tweede en derde normaalvorm

De tweede en derde normaalvorm houden samen in dat iedere attribuut afhankelijk moet zijn van de gehele UID en alleen van de UID. Er mogen dus geen onderlinge afhankelijkheden zijn van andere attributen.

Voorbeeld 1

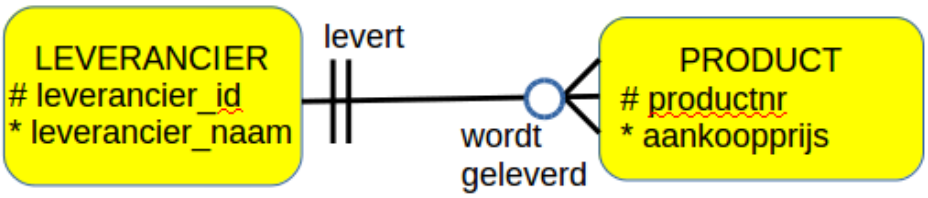
In deze entiteit zijn winkelnaam en winkeladres onderling afhankelijk terwijl zij niet tot de UID behoren. Dat mag niet. Verder zou dezelfde winkelnaam op heel veel rijen voorkomen. Als de winkelnaam zou veranderen zou je dat op heel veel plekken aan moeten passen. De oplossing is een aparte entiteit WINKEL.

Voorbeeld 2:



De entiteit PRODUCT LEVERANCIER heeft de combinatie van kvknr (kamer van koophandel nummer) en productnr als UID. Het probleem is dat leverancier_naam alleen van kvknr afhangt dus niet van de hele UID. Stel dat een leverancier vijf verschillende producten verkoopt en dat op een gegeven moment de leverancier

van naam verandert. Dan moet op vijf verschillende plaatsen de naam van de leverancier aangepast worden. Dat is overbodig werk en foutgevoelig. Zorg er daarom voor dat dezelfde naam nooit meer dan één keer genoteerd hoeft te worden in een database. Oplossing: Maak twee entiteiten: LEVERANCIER en PRODUCT.

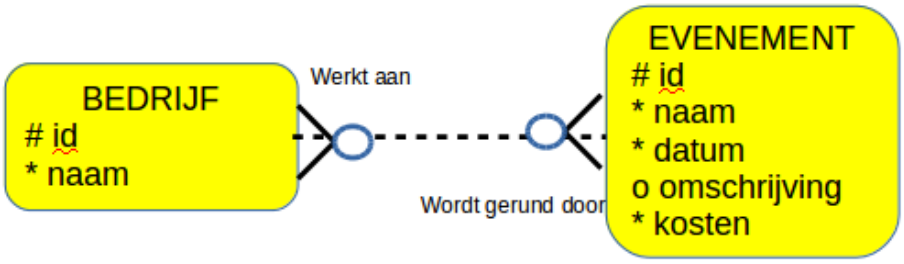


Merk op dat hier een doorgetrokken lijn staat. De UID van PRODUCT is de een combinatie van leverancier_id en product_nr. De eigenschap aankoopprijs is van beide afhankelijk.

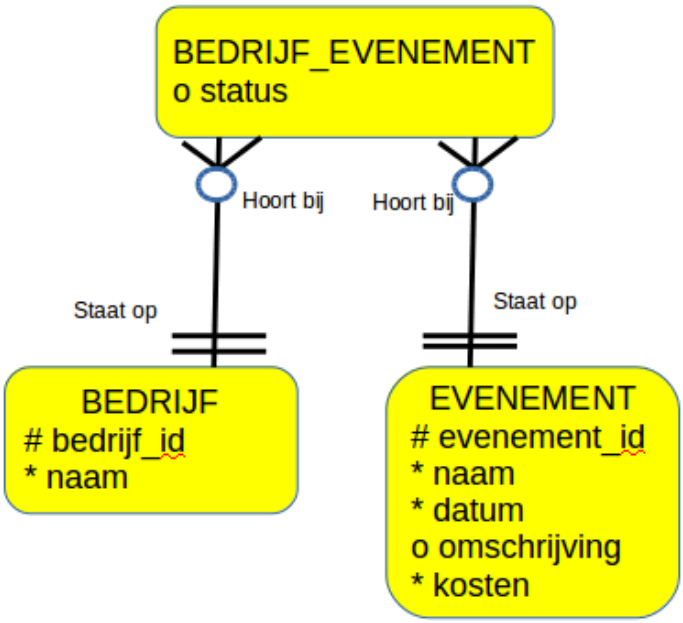
Meer meer relaties

Wanneer je een ERD maakt kom je vaak meer meer relaties tegen. Relationele databases kunnen hier niet mee omgaan. De oplossing is het maken van een intersection entiteit; een tussenentiteit.

Voorbeeld



Oplossing



Je ziet dat de meer meer relatie nu is verdwenen. De lijnen zijn doorgetrokken. Dat betekent dat de UID van BEDRIJF_EVENEMENT een combinatie is van bedrijf_id en evenement_id. De intersection entiteit moet je zien als een soort kaartje. Op ieder kaartje staat een combinatie van een evenement en een bedrijf. Soms, zoals hier, kun je ook nog andere attributen aan de intersection entiteit toevoegen.

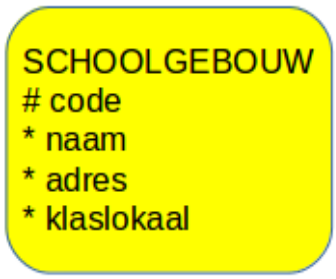
Opgaven

Opgave 1 Garage

Maak een matrix bij de volgende entiteiten: garagebedrijf, auto, persoon.

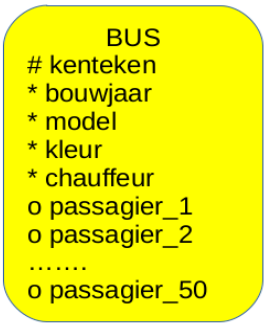
Opgave 2 Schoolgebouw

Normaliseer het volgende ERD:



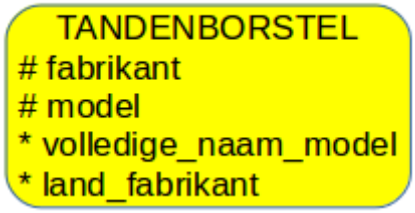
Opgave 3 Bus

Normaliseer de volgende ERD die gaat over een busmaatschappij waarbij de passagiers vooraf via internet een bepaalde lange afstands bus moeten boeken.



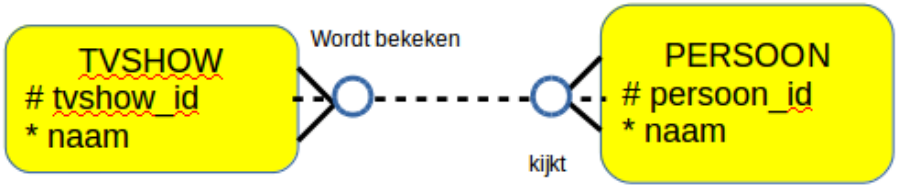
Opgave 4

Normaliseer de volgende ERD:



Opgave 5
Normaliseer de volgende ERD:

Opgave 6
Los de volgende meer meer relatie op.



Hoofdstuk 4 Geschiedenis en supertype subtype

Termen

Geschiedenis

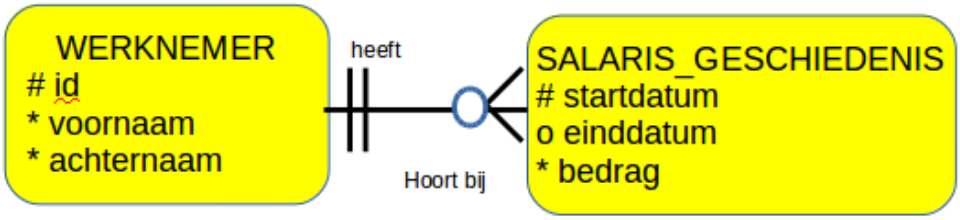
Supertype

Subtype

Geschiedenis

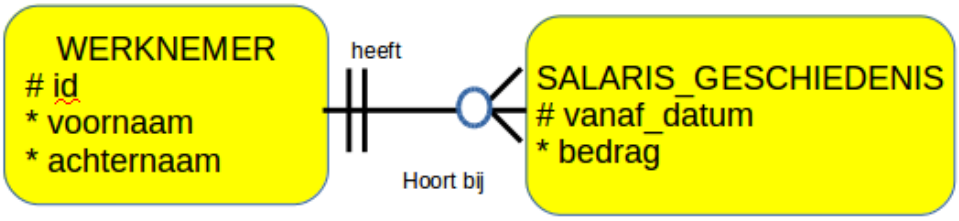
Er zijn veel situaties waarin je iets met tijd moet doen. Hieronder staan twee voorbeelden.

Voorbeeld 1 Salarisgeschiedenis



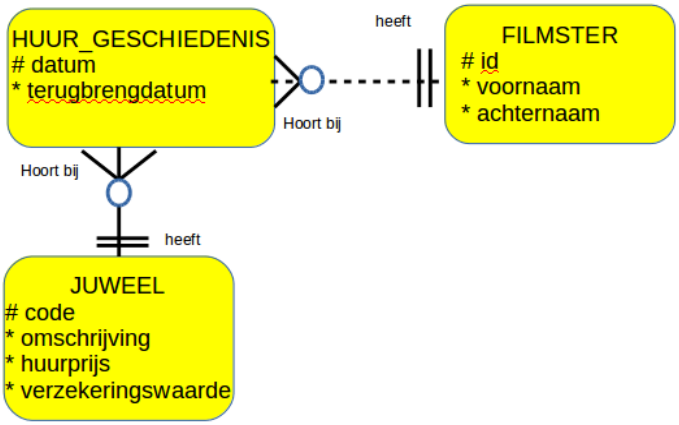
Met behulp van een extra entiteit SALARIS_GESCHIEDENIS kun je het salaris terugzien wat een een werknemer in het verleden heeft verdiend en vanaf welke datum tot welke datum. De einddatum is optioneel omdat bij het verhogen van het salaris nog niet vaststaat tot welke datum de werknemer dit salaris zal krijgen. Merk op dat werknemer_id tot de UID van SALARIS_GESCHIEDENIS behoort want het is een doorgetrokken lijn. Het is nul of één om het makkelijker te maken om het eerste salaris in te voeren.

In de documentatie moet verder een constraint worden opgenomen dat de einddatum niet voor de begindatum kan liggen. Er is een ander ontwerp mogelijk waarbij dit niet nodig is:



Hier werk je niet met een einddatum. Verlaat de werknemer de firma dan zet je het salaris op 0,00.

Voorbeeld 2 Juwelenhuur

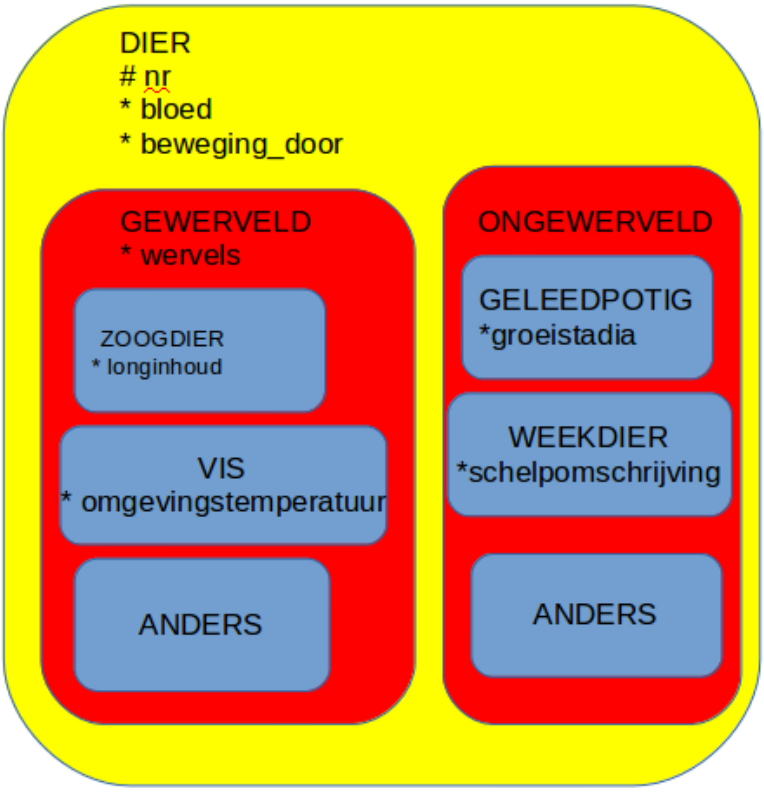


Filmsterren treden vaak op en moeten bij elk optreden andere juwelen dragen. Die juwelen worden niet allemaal gekocht maar vaak gehuurd. Je wilt bijhouden welke filmster welke juwelen heeft gehuurd. De UID van HUUR_GESCHIEDENIS bestaat hier uit datum en juweel_code. Filmster_id behoort niet tot de UID van HUUR_GESCHIEDENIS (stippellijn) omdat, als je dit zou doen, je zou toestaan dat een juweel op dezelfde datum aan twee filmsterren uitgeleend kan worden. Dat gebeurt echter nooit.

Als HUUR_GESCHIEDENIS nog een één-meer relatie zou hebben met een derde entiteit is het handiger om HUUR_GESCHIEDENIS een eigen UID te geven van slechts één attribuut.

Supertype en subtype

Wanneer entiteiten veel dezelfde eigenschappen hebben en een paar verschillende maken we gebruik van de supertype subtype structuur. Hieronder zie je een voorbeeld.



In dit voorbeeld is DIER het supertype en zijn er twee subtypes GEWERVELD en ONGEWERVELD. Deze subtypes bestaan zelf ook weer uit subtypes. GEWERVELD is dus het supertype van VIS. De subtype ANDERS is aangemaakt omdat er nog steeds nieuwe diersoorten worden ontdekt.

Bij supertype subtype gelden een aantal kenmerken:

De subtypes erven alle attributen van het supertype.

In een subtype kunnen opnieuw andere subtypes worden aangemaakt.

Alle exemplaren van het supertype zijn ook exemplaren van één van de subtyes.

Een supertype moet minstens twee subtypes hebben.

Een subtype kan een relatie hebben die een supertype niet heeft.

Fysiek model

Wanneer je dit uitwerkt naar een fysiek model dan maak je voor iedere subentiteit een aparte tabel. Ieder van deze tabellen heeft dan een foreign key naar "id" in de tabel DIER.

Opgaven

Opgave 1

Stel dat er een meerdaagse markt op school gehouden wordt en dat je bij wilt houden wie wanneer welke kraam gaat bemannen. Een kraam wordt maar door één vrijwilliger tegelijk bemand. Sommige vrijwilligers kunnen langer werken dan anderen. Het schema moet van tevoren worden gemaakt, zodat bepaald kan worden wanneer de kraam nog niet bemand is. Maak een ERD bestaande uit drie entiteiten voor deze situatie.

Opgave 2

Noem minimaal drie constraints die apart geprogrammeerd moeten worden bij het informatiesysteem uit de vorige opgave.

Opgave 3 Kledingzaak

Onze zaak verkoopt verschillende soorten vrouwenkleren: jurken, shirts en blouses.
Ie der product heeft een naam, omschrijving en een prijs. Alle producten hebben ook een taillemaat. Jurken en shirts hebben een lengte maar blouses niet. Jurken en en blouses hebben een bustemaat maar shirts niet.

a) Welke entiteiten zitten in dit verhaal?
b) Welke entiteit is het supertype?
c) Welke attributen behoren tot het supertype?
d) Welke UID heeft het supertype?
e) Noteer bij ieder subtype de attributen.
f) Teken de ERD.

Hoofdstuk 5 Hiërarchische en Recursieve relaties

Termen

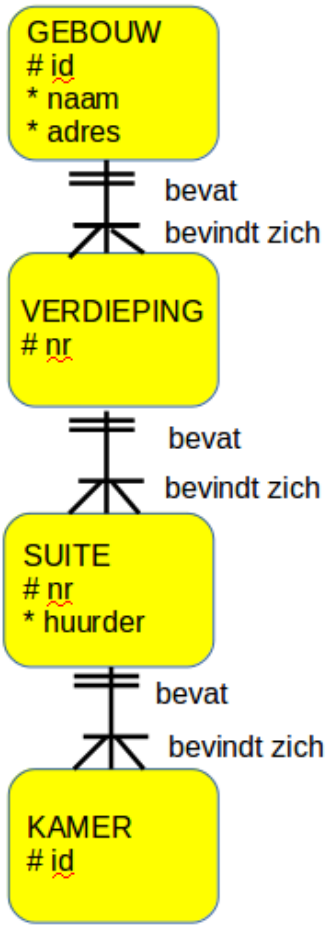
Hierarchisch

Recursief

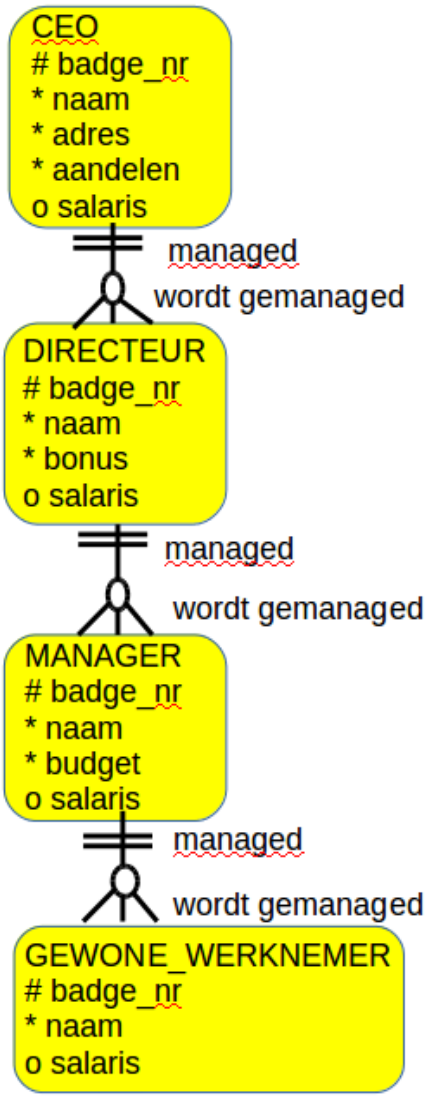
Hiërarchische relaties

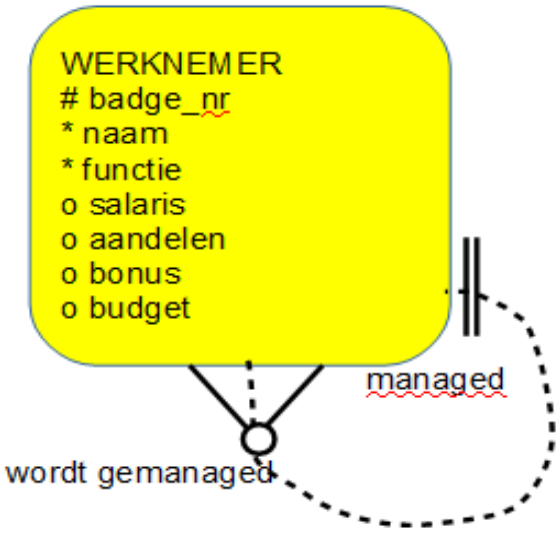
Hiernaast zie een een hiërarchische relatie. Deze wordt zo genoemd omdat de onderliggende entiteiten de UID van de entiteit er boven erven.

Hiërarchishe relaties worden vooral gebruikt voor structuren waarvan je verwacht dat ze niet snel zullen veranderen. Een database met een hiërarchische relatiestructuur is namelijk lastig aan te passen. Als men bijvoorbeeld zou besluiten om niet meer met SUITES te werken klopt er van de hele database niets meer.



Recursieve relaties





Links zie je de organisatie van de werknemers volgens een hiërarchische structuur. Deze structuur is echter niet aan te bevelen omdat reorganisaties bij bedrijven veel voorkomen. Het zou zomaar voor kunnen komen dat er een laag uitgehaald wordt en dat er volgend jaar geen MANAGERS meer zijn. Dan heb je met een hiërarchische structuur een probleem.

Ernaast zie je het flexibeler alternatief; alles in één entiteit WERKNEMER met wat optionele eigenschappen. In deze figuur zie je ook een **recursieve** relatie. Dit is een relatie van een entiteit met zichzelf.

Iedere WERKNEMER managet nul of meer WERKNEMERS.

Iedere WERKNEMER wordt gemanaged door precies één WERKNEMER.

Opgaven

1. Wat is de UID van KAMER in het voorbeeld van de hiërarchische relaties?
2. Welke relatie in het voorbeeld van de hiërarchische relaties is niet overdraagbaar?
3. Noem een voordeel van een recursieve relatie structuur t.o.v. een hiërarchische relatiestructuur in het voorbeeld over de organisatie van de werknemers.
4. In het voorbeeld van de organisatie van de werknemers kent de rechter structuur veel optionele attributen. Met welke andere structuur zou je dit kunnen verminderen? Teken de ERD.