II. BASISBEGRIPPEN

Het semantisch model is een datamodel dat logisch volgt op het relationele model. De kritieken op het relationele model werden geformuleerd door het echtpaar Smith in [2] en [3]. Zij probeerden de door hen aangegeven tekortkomingen op te lossen. Dit alles leidde na verdere aanpassingen tot het semantisch model, waarvan hieronder de belangrijkste begrippen worden besproken. Voor een uitvoeriger beschrijving wordt verwezen naar [1].

II.1. (BASIS)TYPEN

In het semantisch model onderscheidt men **basistypen** (zonder attributen) en **samengestelde typen** (met attributen). Basistypen kunnen bijvoorbeeld zijn: naam, adres, woonplaats, geboortedatum. Uit deze basistypen kan men de samengestelde typen maken. Zo zou men uit de genoemde basistypen het type student kunnen maken. We schrijven dat als volgt op:

type student = naam, adres, woonplaats, geboortedatum.

Bovenstaande typedefinitie kunnen we associëren met het werkwoord HEBBEN, immers, een student heeft een naam, adres, woonplaats, en een geboortedatum.

De voorkomens van een type worden **instances** genoemd. Iedere instance krijgt een **identificatie**. Zo kunnen bijvoorbeeld de drie instances van het type student voorkomen die in de volgende figuur gegeven zijn:

identificatie student	naam	adres	woonplaats	geboortedatum
123456	P. Stavast	Straatweg 5	Amsterdam	01-05-1960
312645	W. Peters	Peensteeg 7	Rotterdam	05-01-1965
981265	F. Zomer	Keetweg 1	Amsterdam	16-11-1950

Voorbeeld instances van type student

Merk op dat de identificatie door een dubbele streep van de beschrijvende attributen wordt gescheiden. Dit zal in het vervolg ook zo gedaan worden.

Een type wordt bepaald door zijn attributen (naam, adres, etc.). Dit in tegenstelling tot het relationele model waar een relatie bepaald wordt door haar elementen (’tuples’). De invarianten van de werkelijkheid zijn echter de attributen en niet de instances. Zo kan de verzameling werknemers (de elementen) van een bedrijf veranderen, maar de attributen van een werknemer (bijvoorbeeld naam, adres, woonplaats, afdeling) blijven invariant. Het begrip

type is dus een betere basis voor een datamodel dan het wiskundige begrip relatie.

Een samengesteld type wordt altijd uit typen (samengestelde- of basistypen) gevormd. Men noemt de typen waaruit een nieuw type wordt gemaakt de **attributen** of **eigenschappen** van het type.

Soms komt het voor dat een type verschillende attributen heeft die van hetzelfde type zijn. Bijvoorbeeld:

type artikel = naam, gewicht, prijs, prijs.

Hier komt twee keer het type prijs voor als attribuut van het type artikel. De eerste prijs kan de inkoopprijs zijn, terwijl de tweede prijs de verkoopprijs kan voorstellen. Om in dit soort gevallen verwarring te voorkomen is het mogelijk om met **rolnamen** te werken. Zo'n rolnaam wordt als prefix voor de typenaam geplaatst gescheiden door een underscore ('_'). Men kan dus definiëren:

type artikel = naam, gewicht, inkoop_prijs, verkoop_prijs.

Een attribuut waarvan de aanduiding begint met een rolnaam wordt een **rolattribuut** genoemd. In een typedefinitie mag een attribuutnaam nooit meer dan 1 keer voorkomen. Immers een type heeft wel of niet een bepaalde eigenschap, maar een type kan niet meerdere keren een zelfde eigenschap hebben. Men kan overigens ook nog een andere oplossing kiezen met verkoopprijs en inkoopprijs als basistypen. In verband met betekenis (semantiek) en aantal basistypen valt deze keuze af te raden.

In het semantisch model kan men op twee manieren typen maken uit (basis)typen, namelijk door **aggregatie** en door **generalisatie**. Op deze wijze kan men een **abstractiehiërarchie** opbouwen van typen.

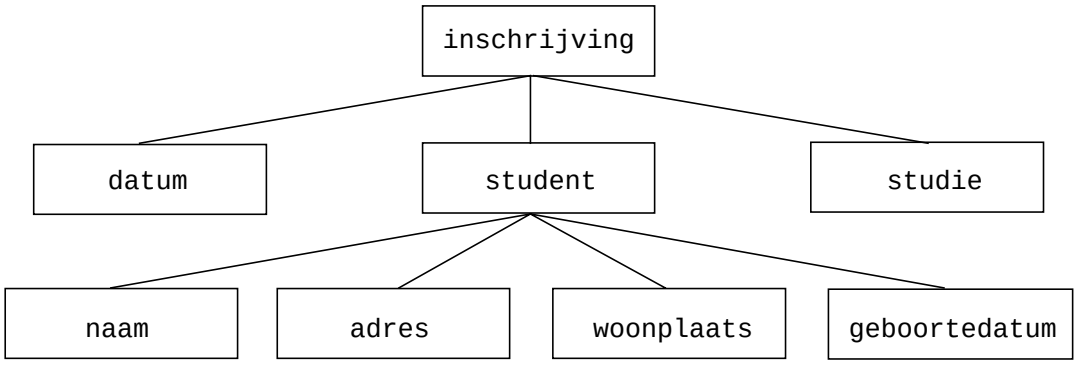
II.2. AGGREGATIE

Een aggregatie is een benoemde relatie tussen typen. In het voorbeeld werden de basistypen naam, adres, woonplaats en geboortedatum geaggregeerd tot een nieuw type. Dit nieuwe type werd student genoemd en kan op een hoger niveau als ondeelbaar worden beschouwd.

Laten we nu het voorbeeld uitbreiden. Stel dat een student zich op een bepaalde datum inschrijft voor een bepaalde studie. We kunnen dan het model als volgt definiëren:

type student = naam, adres, woonplaats, geboortedatum.
type inschrijving = datum, student, studie.

Dit model kan men ook als volgt weergeven:



Aggregatie

Merk bij deze figuur op dat de lijnen lopen van het midden van de onderkant van de aggregatie naar het midden van de bovenkant van de (basis)typen waaruit de aggregatie bestaat.

Uit dit voorbeeld volgt dus dat ieder type dat een aggregatie van (basis)typen is, zelf ook weer een deel van een aggregatie op hoger niveau kan zijn. Op het niveau van inschrijving wordt de student als ondeelbaar beschouwd. Afhankelijk van het beschouwingsniveau wordt student gezien als attribuut of als type. Deze uitwisselbaarheid van begrippen is een essentieel kenmerk van het semantisch model.

De inverse bewerking van aggregeren wordt **decompositie** (ontleden) genoemd. Zo kunnen we het type inschrijving decomponeren in het type student en de basistypen datum en studie.

II.3. GENERALISATIE

Een generalisatie is een abstractie, waarbij een aantal typen bepaalde eigenschappen gemeen hebben en waarbij men deze gemeenschappelijke combinatie van eigenschappen voor een bepaalde toepassing als één objecttype wenst te beschouwen. Het nieuwe type wordt de generalisatie genoemd.

Iedere instance van de oorspronkelijke typen verwijst naar precies één instance in de generalisatie. Twee instances van verschillend type die op deze wijze bij elkaar horen kunnen echter verschillende namen (identificaties) hebben. Het worden daarom **synoniemen** genoemd. Stel bijvoorbeeld dat we de volgende definities hebben:

type hoogleraar = naam, adres, woonpl, indiensttredings_datum, vakgebied.
type secretaresse = naam, adres, woonpl, indiensttredings_datum, diploma.

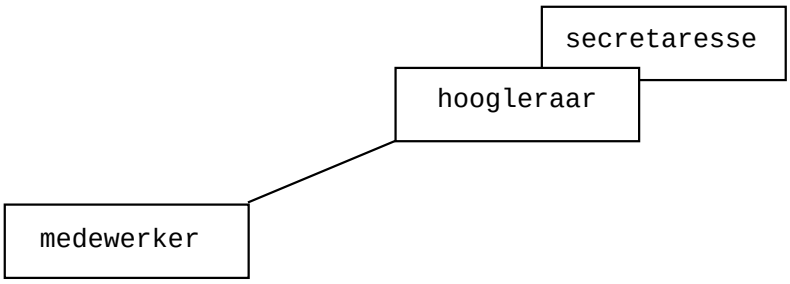
We zien dat in deze typen de eerste vier attributen gelijk zijn. We kunnen dus de typen hoogleraar en secretaresse **generaliseren** (natuurlijk maken we deze generalisatie alleen als

deze nuttig is voor ons model). De naam van de generalisatie zou medewerker kunnen zijn. Iedere hoogleraar is (verwijst naar) een medewerker en ook iedere secretaresse is een medewerker. De definities worden dus:

type medewerker = naam, adres, woonpl, indiensttredings_datum.
type hoogleraar = [medewerker], vakgebied.
type secretaresse = [medewerker], diploma.

Generalisatie kunnen we ook associëren met het werkwoord ZIJN, immers, een secretaresse is een medewerker en een hoogleraar is eveneens een medewerker.

De rechte haken om medewerker geven aan dat het hier om een generalisatie in plaats van een aggregatie gaat. Dit model kunnen we weergeven als in onderstaande figuur.



Generalisatie

Bij de generalisatie tekenen we de lijnen van hoek naar hoek, waarbij de generalisatie (type medewerker) onder staat. De typen hoogleraar en secretaresse worden de **categorieën** van het type medewerker genoemd. De generalisatie van A en B wordt in [1] als volgt gedefinieerd: de eigenschappen die zowel in A als in B voorkomen.

Natuurlijk kan ook de generalisatie gemaakt worden van meer dan twee typen. Een eigenschap hoort dan tot de generalisatie als die eigenschap tot alle typen behoort.

Het omgekeerde van generaliseren wordt **specialiseren** genoemd. Ook de specialisatie wordt in [1] gedefinieerd: eigenschappen die voorkomen in A of in B.

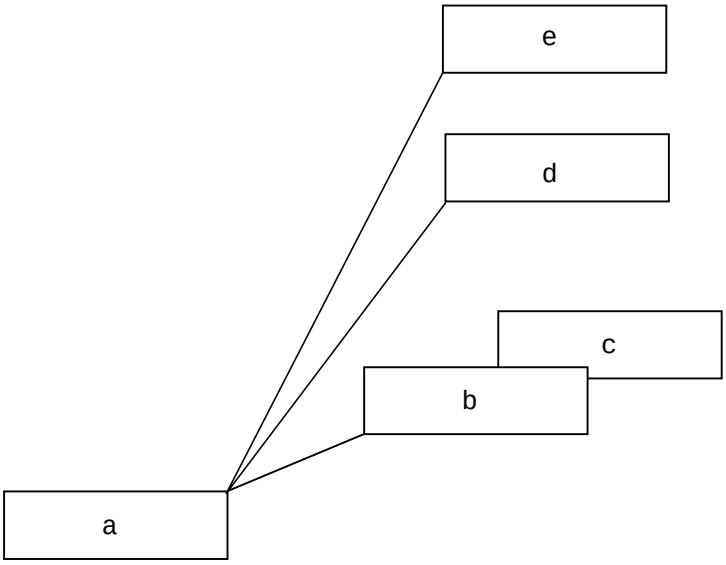
Ook hier kan men natuurlijk weer de specialisatie maken van meer dan twee typen. De specialisatie bevat dan alle eigenschappen die in de oorspronkelijke typen voorkomen. Bovendien bevat een specialisatie minstens één eigenschap die niet in de generalisatie voorkomt. Anders is er geen reden voor onderscheid tussen specialisatie en generalisatie.

Er kunnen natuurlijk medewerkers zijn die geen hoogleraar en ook geen secretaresse zijn. In het algemeen hoeft het dus niet zo te zijn dat ieder instance van een generalisatie hoort bij een instance uit één der categorieën. Wel moeten de categorieën onderling disjunct (gescheiden) zijn, dat wil zeggen dat instances uit verschillende categorieën niet naar dezelfde

instance van de generalisatie mogen verwijzen. Er mogen dus geen medewerkers voorkomen die zowel hoogleraar als secretaresse zijn. Komen zulke personen toch voor in de werkelijkheid, dan moet het model worden aangepast.

De specialisaties behoren altijd tot **blokken**. De specialisaties **binnen** een blok zijn dan onderling disjunct. Als er meer dan één blok is, moet opgegeven worden tot welk blok de betreffende specialisatie behoort. Ook hier moet men dus met rolnamen werken. Het model van onderstaande figuur wordt op die manier als volgt gedefinieerd:

type a = ...
type b = [x_a], ...
type c = [x_a], ...
type d = [y_a], ...
type e = [z_a], ...



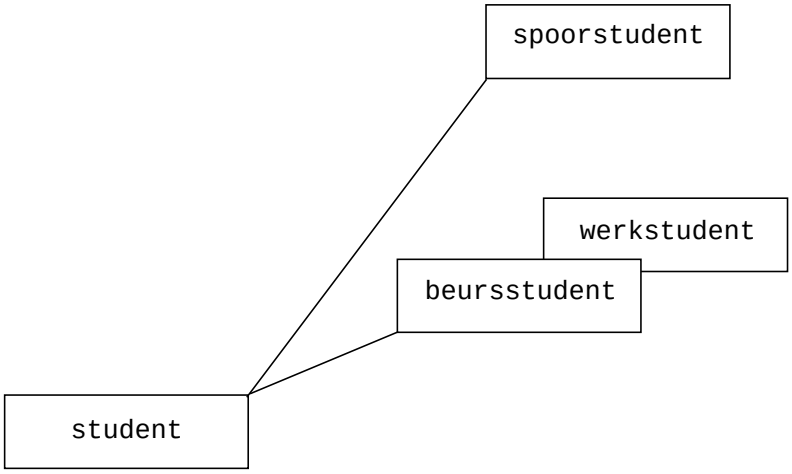
Specialisaties in blokken

In dit voorbeeld hoeven dus slechts de categorieën b en c onderling disjunct te zijn. De typen b en c behoren tot één blok. Dit is afleidbaar uit de typedefinities omdat voor b en c de generalisatie a van dezelfde prefix "x" is voorzien.

Het bovenstaande kan met het volgende voorbeeld worden toegelicht:

type student = ...
type spoorstudent = [student], dagkilometers.
type werkstudent = [bijzonder_student], werkuren.
type beursstudent = [bijzonder_student], beursbedrag.

Hierbij is verondersteld dat een student niet tegelijk tot de categorieën werkstudent en beursstudent kan behoren. Deze beide typen worden als disjunct beschouwd en behoren tot het blok dat met "bijzonder" kan worden aangeduid. Zowel voor beursstudenten als voor werkstudenten geldt dat ze tegelijkertijd ook spoorstudent kunnen zijn. Spoorstudent vormt nu een apart blok van specialisatie dat niet disjunct is van het blok "bijzonder". Deze situatie wordt als volgt aangegeven:



Voorbeeld van niet-disjuncte specialisaties

Stel dat er een student "123456" bestaat dan zijn volgens het genoemde model de volgende instances toegestaan:

spoorstudent	[student]	dagkilometers
78934	123456	98

beursstudent	[bijzonder_student]	beursbedrag
59677	123456	9976

Echter één of beide specialisaties hoeven niet voor elke student te bestaan.

De instances in een specialisatie vormen een subset van de instances in de bijbehorende generalisatie (vanwege relateerbaarheid). Echter de vereniging van een aantal specialisaties van hetzelfde blok hoeven dus niet precies de verzameling in de generalisatie op te leveren.

Het bestaan van de eerder genoemde instance (voorkomen) van beursstudent sluit uit het mogen bestaan van een instance van werkstudent gebaseerd op dezelfde student "123456". Immers beursstudent en werkstudent zijn disjuncte categorieën.

II.4. CONCEPTUEEL MODEL

Alle typedefinities bij elkaar vormen nu een model van de werkelijkheid. Dit noemt men het conceptuele model van de database. De database is in principe ontworpen voor het opslaan van gegevens over dat deel van de werkelijkheid dat voor een bepaalde toepassing relevant is. Het conceptuele model dient dus ook als afbeelding van een bepaalde informatiebehoefte.

Met de abstracties aggregatie en generalisatie kan men verbanden in het model aanbrengen. We spreken echter pas van een **semantisch model** als het model aan de eisen van **omkeerbaarheid** (convertibility) en **relateerbaarheid** (relatibility) voldoet. Deze twee eisen, waaraan zowel op typeniveau als op instance-niveau voldaan moet zijn, worden hieronder besproken.

II.5. OMKEERBAARHEID

Op typeniveau moet gelden:
Ieder type wordt slechts één keer gedefinieerd en twee verschillende typen hebben niet dezelfde definitie. Deze twee eisen op typeniveau worden door Xplain bewaakt.

Men mag dus niet definiëren:

type student = naam, adres, woonplaats, geboorte_datum.
type student = [persoon], studierichting.

Maar ook de volgende definities zijn niet goed:

type student = naam, woonplaats, geboorte_datum.
type persoon = naam, woonplaats, geboorte_datum.

Op instance-niveau moet gelden:
Binnen een type mag een identificatie slechts éénmaal gebruikt worden en twee instances (met verschillende identificaties) mogen niet voor alle attributen dezelfde waarden hebben. Deze eis wordt door Xplain bewaakt.

De instances uit de onderstaande figuur voldoen dus niet aan deze eis omdat de identificatie "A001" twee keer voorkomt:

persoon	naam	adres	woonplaats	geboorte_datum
A001	P. Stavast	Straatweg 5	Amsterdam	01-05-1960
A001	W. Peters	Peensteeg 7	Rotterdam	05-01-1965

Instances die niet voldoen aan de omkeerbaarheid

wegvoer-tuig	merk	handels-aanduiding	wielbasis
1004	DAF	FT1600	265
1008	DAF	FT1600	265

Instances die niet voldoen aan de omkeerbaarheid

Deze eis wordt niet door Xplain bewaakt. De ontwerper dient zich bij ontwerp ervan te gewissen (door de opname van voldoende attributen) dat deze situatie niet kan voorkomen.

Anders wordt dit laatste voorbeeld als ook nog gegeven was de definitie:

type motorvoertuig = [wegvoertuig], cilinderinhoud, vermogen.

en de volgende instances bestaan:

motor-voertuig	[wegvoer-tuig]	cilinder-inhoud	vermogen
M410	1004	6170	153
M412	1008	5800	140

Instances

In dat geval mogen de instances uit het voorbeeld van de voorlaatste figuur wel voorkomen. De omkeerbaarheid moet namelijk alléén gelden voor de **volledige** beschrijving van een object en hoeft niet te gelden voor een deel ervan.

Opgemerkt moet worden dat instances van verschillende typen wel dezelfde identificatie mogen hebben. Men noemt dit dan **homoniemen**. Zo kan een motorvoertuig "2058" als identificatie hebben, terwijl "2058" ook als identificatie van een bepaalde student mag bestaan. Ook kan het voorkomen dat een bepaald object uit de werkelijkheid tot verschillende typen behoort (bijvoorbeeld student, persoon en lid van een bibliotheek). In deze verschillende typen kan het object dan verschillende identificaties hebben. Dit worden **synoniemen** genoemd. Zo is in deel II.3 een bepaalde secretaresse synoniem met een bepaalde medewerker.

II.6. RELATEERBAARHEID

Voor de relateerbaarheid op typeniveau moet gelden:
Alle attributen in een typedefinitie zijn gedefinieerd als basistype of als samengesteld type. Deze eis wordt door Xplain bewaakt.

Op instance-niveau moet voor de relateerbaarheid gelden:
Voor iedere instance moet de waarde van een attribuut, dat zelf een samengesteld type is, voorkomen als identificatie van een instance van dat type. Ook deze eis wordt door Xplain bewaakt.

Stel bijvoorbeeld dat we de volgende definities hebben:

type inschrijving = datum, student, studie.
type student = naam, adres, woonplaats, geboorte_datum.

Een instance uit de volgende twee figuren voldoet niet aan de relateerbaarheid:

inschrijving	datum	student	studie
I235	05-01-1973	A002	Informatica
I237	06-10-1980	A001	Wiskunde
I239	09-11-1976	A003	Wiskunde
I271	10-08-1972	A002	Mechanica

Instance die niet aan de relateerbaarheid voldoet

student	naam	adres	woonplaats	geboorte_datum
A001	P. Stavast	Straatweg 5	Amsterdam	01-05-1960
A002	W. Peters	Peensteeg 7	Rotterdam	05-01-1965

Student "A003" is niet aanwezig binnen de database

Het is nu de vraag wie student "A003" van inschrijving "I239" is. Dit soort verwijzingen mag niet voorkomen. De verzameling van studenten die bij een inschrijving voorkomt moet dus op ieder moment een deelverzameling zijn van de verzameling van studenten. De eis van relateerbaarheid op instance-niveau wordt daarom ook wel aangeduid als de **subset-invariant**.

II.7. ONTLEDINGSBOOM

In een semantisch model wordt de samenhang tussen de typen aangegeven met behulp van de abstracties aggregatie en generalisatie. Aldus ontstaat een zogeheten abstractiehiërarchie.

Voor de gebruikers van een database is het nuttig eens te bekijken hoe deze hiërarchie zich openbaart op instanceniveau.

Vanaf dit punt in de handleiding zal bij voorbeelden voornamelijk gebruik worden gemaakt van het model van de warenhuis database dat beschreven wordt in het hoofdstuk X.1.

Waar de gebruiker van een database uiteindelijk in is geïnteresseerd zijn de opgeslagen gegevens, de instances van de verschillende typen. Deze instances staan soms met elkaar in verband. Welke vormen van samenhang we hierbij onderscheiden, wordt in deze paragraaf beschreven.

Een eerste samenhang is de samenhang die bestaat tussen instances van hetzelfde type. Zo zijn de volgende twee instances beide van het type afdeling en beschrijven dus twee objecten uit de werkelijkheid, met dezelfde attribuuttypen.

afdeling	afdnaam	verdieping	telnr	chef_naam
22	meubel	2	2244	Opmeer
34	sportartikelen	1	3156	Jans

Belangrijker echter is de samenhang die bestaat tussen instances van verschillende typen. We beschouwen de volgende instance van het type verkart, een afkorting van verkoopartikel.

verkart	artikel	afdeling
499	1730037	34

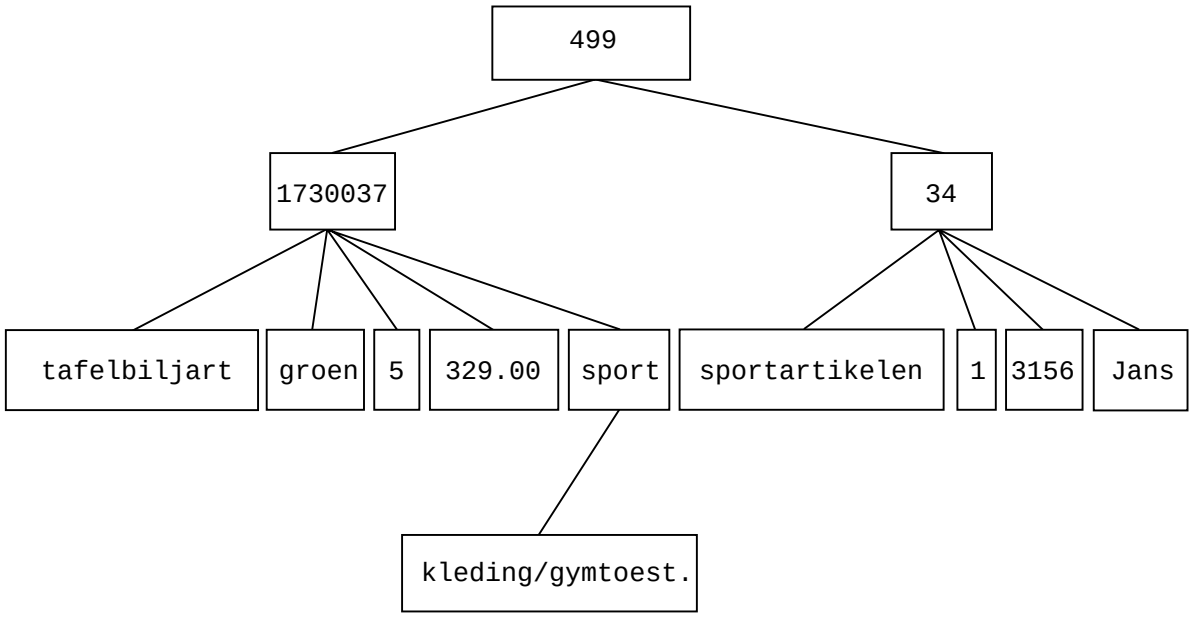
Het type verkart heeft twee attributen en een instance van dit type bestaat dus uit twee attribuutwaarden en een identificatie. Beide attributen zijn echter zelf ook samengestelde typen, zodat de attribuutwaarden identificaties zijn van respectievelijk een instance van het type artikel en een instance van het type afdeling. Deze identificaties staan op hun beurt weer voor een reeks attribuutwaarden, bijvoorbeeld:

artikel	beschrijving	kleur	voorraad	prijs	soort
1730037	tafelbiljart	groen	5	329.00	sport

afdeling	afdnaam	verdieping	telnr	chef_naam
34	sportartikelen	1	3156	Jans

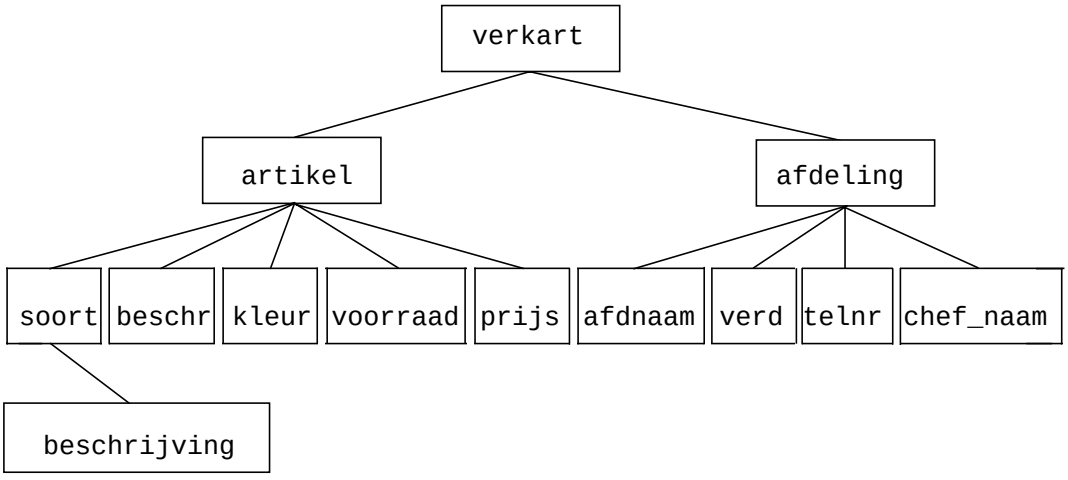
De instance van artikel bevat weer een identificatie van een soort-instance:

soort	beschrijving
sport	kleding/gymtoest.



De ontledingsboom van instance 499 van verkart.

Aldus ontstaat er een instance-structuur die het beste is weer te geven met behulp van een boom. In het vervolg wordt gesproken over de **ontledingsboom van een instance**. Alle attribuutwaarden in deze ontledingsboom hebben betrekking op die ene instance van het type verkart (verkoopartikel).



De ontledingsboom van het type verkart

We kunnen bladeren en knopen onderscheiden in een ontledingsboom:

- in een blad staat een basistype (niet verder ontleedbaar);

- in een knoop staat een samengesteld type.

In een ontledingsboom zijn de waarden in de bladeren attribuutwaarden van attributen die basistypen zijn en de waarden in de knopen zijn identificaties van instances van een samengesteld type.

Voor elke instance in de database is een dergelijke ontledingsboom op te stellen. Het zal direct duidelijk zijn dat de ontledingsbomen van instances van eenzelfde type een identieke structuur hebben. Voor de instances van het type verkart is deze algemene structuur in de vorige figuur weergegeven. We spreken over de ontledingsboom van een type; in dit geval van het type verkart. Zo'n ontledingsboom is natuurlijk niets anders dan een gedeelte van de abstractiehiërarchie en wel het gedeelte dat in de hiërarchie onder het betreffende type ligt. Hierbij zij opgemerkt, dat het bovenstaande onverkort geldt in gevallen waarbij een type een specialisatie is van één of meer van zijn attributen. Zo zal in de ontledingsboom van het type spoorstudent onder spoorstudent staan: [student] en dagkilometers.

De **ontledingsboom van een type** is dus een weergave van de toegepaste ontledingen en generalisaties uitgaande van dit type.

II.8. NESTING

We kunnen ook de samenhang beschouwen tussen een instance van een bepaald type en instances van typen op een hoger gelegen niveau in de hiërarchie. We gaan daarbij uit van de volgende afdeling-instance:

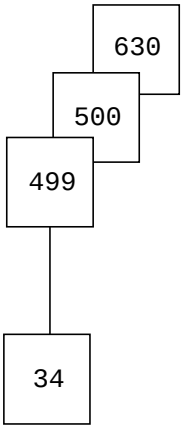
afdeling	afdnaam	verdieping	telnr	chef_naam
34	sportartikelen	1	3156	Jans

Het type afdeling is opgenomen als attribuut bij het type verkart, een aggregatie die aangeeft welke artikelen er op welke afdelingen verkocht worden. Er is een aantal instances van het type verkart die afdeling 34 als attribuutwaarde heeft, bijvoorbeeld:

verkart	artikel	afdeling
499	1730037	34
500	1730036	34
630	1443191	34

Dit aantal instances is echter niet voor iedere afdeling gelijk, is tijdsafhankelijk en kan zelfs nul zijn. Het is een variabele eigenschap van een afdeling en ligt daarom niet vast in het model. Het aantal instances van verkart horende bij afdeling 34 kan echter wel door telling bepaald worden.

Een dergelijke combinatie van instances, als bovenstaande afdeling-instance met de verkart-instances die ernaar refereren, wordt een **groep** genoemd. In dit verband wordt de afdeling-instance de onderliggende instance en worden de verkart-instances de leden van de groep genoemd. Een groep wordt weergegeven als volgt:

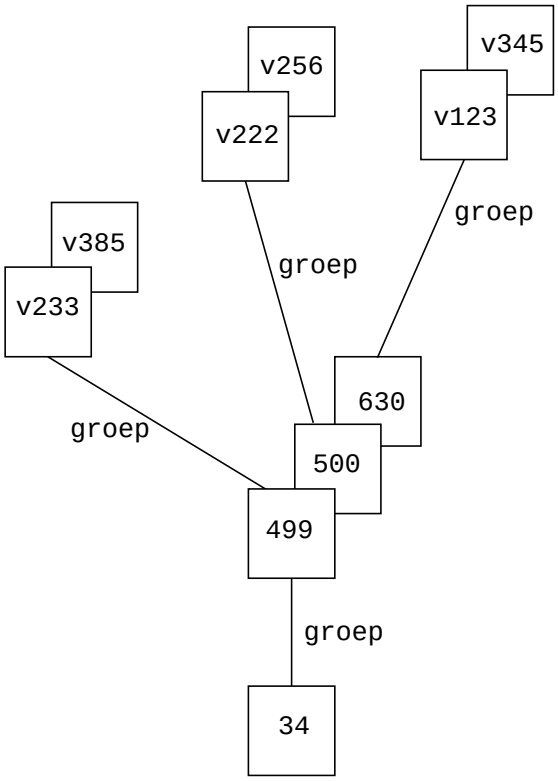


Voorbeeld van een groep

Het type verkart is op zijn beurt opgenomen als attribuut van het type verkoop, zodat elke instance van het type verkart onderliggende instance is van een groep met leden van het type verkoop. De situatie in ons voorbeeld is bijvoorbeeld als in de volgende figuur. Dit principe kan worden voortgezet. In het voorbeeld is dit echter niet het geval, daar het type verkoop bij geen enkel ander type als attribuut is opgenomen (zie figuur blz. 34).

Er is echter nog een aspect. Het type afdeling is, naast attribuut bij het type verkart, ook nog attribuut bij het type werknemer. Dus iedere afdeling-instance is tevens onderliggende instance van een groep met instances van het type werknemer. Voor afdeling 34 is de situatie als in de volgende figuur.

Aldus ontstaat er een nesting van groepen, uitgaande van die ene afdeling-instance. Dit wordt de **nesting van een instance** genoemd. Ook nu geldt dat er voor iedere instance uit de database een nesting is op te stellen en dat de nestings van instances van hetzelfde type een identieke structuur hebben. Voor het type afdeling is deze algemene structuur weergegeven in de laatste figuur van deze paragraaf.

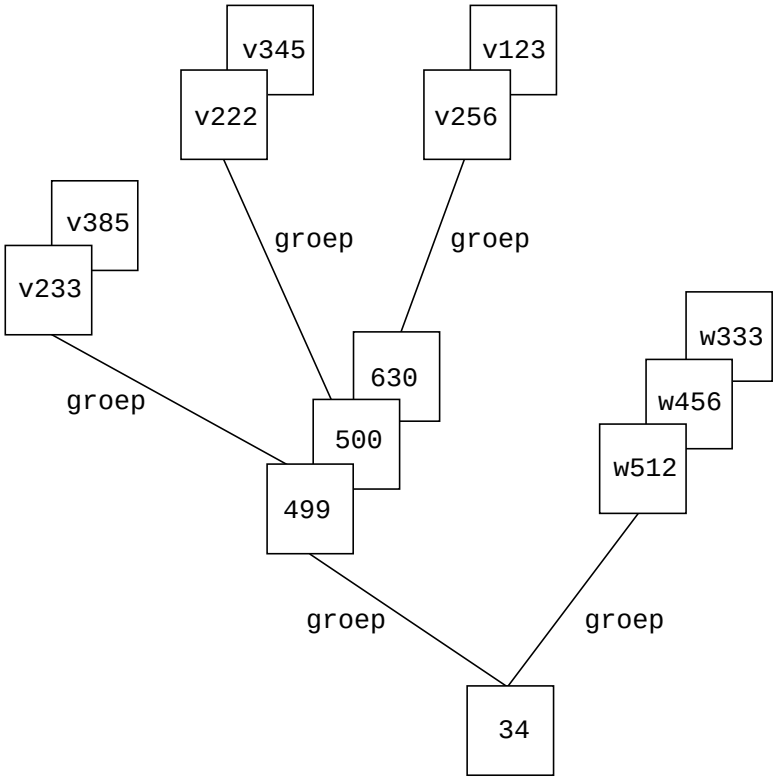


Voorbeeld van een groepstructuur

We spreken over de **nesting van een type**, in dit geval van het type afdeling. Het type afdeling bevindt zich op niveau 1 in de nesting, de typen verkart en werknemer op niveau 2, enzovoort. De nesting van het type afdeling bestaat uit twee takken:
de tak /afdeling/groep verkart/groep verkoop en
de tak /afdeling/groep werknemer/.

De notatie spreekt voor zich. Een dergelijke vertakking kan natuurlijk ook op een hoger niveau voorkomen.

Ook de nesting van een type is niets anders dan een gedeelte van de abstractiehiërarchie en wel het gedeelte dat boven het betreffende type ligt. Hierbij zij opgemerkt dat het bovenstaande onverkort geldt in gevallen waarbij het onderliggende type een generalisatie is; het enige verschil is, dat de specialisatiegroep hoogstens één instance mag bevatten.



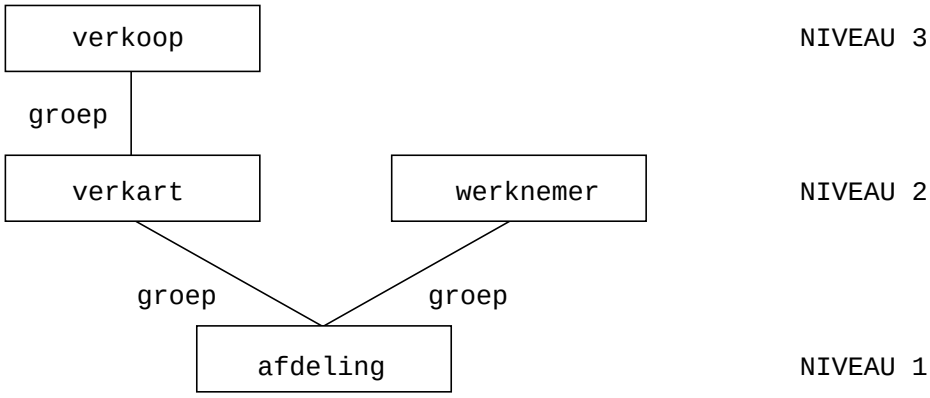
Voorbeeld van de nesting van instance 34 van afdeling

De nesting van een type is dus een weergave van de toegepaste aggregaties en specialisaties uitgaande van dit type.

Er is een wezenlijk verschil tussen de ontledingsboom en de nesting van een type. De ontledingsboom geeft de vaste eigenschappen van het type weer, immers elke instance van het type heeft een identieke ontledingsboom, slechts de betreffende waarden verschillen. De nesting van een type daarentegen, geeft de variabele eigenschappen van het type weer, immers voor elke instance kan het aantal leden van de groepen anders zijn en indien voor een bepaalde groep dit aantal nul is, bestaan de groepen op hogere niveaus niet. Zo zullen er geen verkopen op een afdeling plaatsvinden als er geen artikelen te koop zijn op die afdeling.

Er is nu enig inzicht verschaft in de samenhang die bestaat tussen een bepaalde instance uit de database en de overige instances uit de database. Het is deze samenhang waar de gebruiker van een database in geïnteresseerd is.

Later zal blijken dat de hier ingevoerde begrippen ontledingsboom en nesting een grote steun zijn bij de introductie van de datamanipulatietaal en de panelgenerator.



De nesting van het type afdeling