Normalisatie

Leereenheid 3

Normalisatie

INTRODUCTIE

Het correct zijn van een relationele databasestructuur hangt nauw samen met de betekenis van de gegevens. Dit hebben we al gezien bij de bespreking van *redundantie* en *normalisatie*, in Inleiding informatica, leereenheid 11 'Relationele databases: structuur'. Een structuur die redundantie toelaat, is onwenselijk, tenzij dit een bewuste keuze is en er databaseregels zijn die inconsistente gegevensopslag voorkomen. Normalisatie is een vorm van *schematransformatie*: het ene relationele *databaseschema* (beschrijving van structuur en regels) wordt getransformeerd naar het andere. De bedoeling is dat het eindresultaat voldoet aan het principe 'elk entiteittype zijn eigen tabel'. We hebben echter gezien dat dit niet altijd het geval is: via standaardisatie moeten sommige entiteittypen achteraf alsnog boven tafel worden gehaald.

Normaliseren kán onderdeel zijn van het modelleer- en ontwerpproces, waarbij op basis van een model van een 'relevante werkelijkheid' een databaseschema wordt afgeleid. Tegenwoordig echter komt men meestal langs een andere weg tot de juiste, genormaliseerde databasestructuur. Maar de normalisatietheorie kan dan nog steeds worden gebruikt als *controlemiddel* en levert ook de *taalmiddelen* om over een goede structuur te spreken. Normalisatie is en blijft daardoor een belangrijk onderdeel van de relationele theorie. In deze leereenheid gaan we nog een stapje dieper dan in Inleiding informatica.

LEERDOELEN

Na het bestuderen van deze leereenheid wordt verwacht dat u

- kanttekeningen kunt plaatsen bij het verbod op herhalende groepen (de eerste normaalvorm)
- het begrip functionele afhankelijkheid kunt omschrijven in termen van de semantiek van een database
- kunt aangeven wanneer een relationele structuur ongewenst is, gezien een bepaalde functionele-afhankelijkheidsrelatie
- de hogere normaalvormen (tweede, derde, Boyce-Codd) kunt kenschetsen in termen van specifieke 'verbodsbepalingen' op bepaalde typen redundantie
- elke hogere normaalvorm kunt illustreren via een 'kritisch' voorbeeld
- elke niet volledig genormaliseerde structuur kunt normaliseren tot de derde normaalvorm
- eenvoudige niet volledige genormaliseerde structuren kunt normaliseren tot de Boyce-Codd-normaalvorm.

De studielast van deze leereenheid bedraagt 5 uur.

LEERKERN

1 De eerste normaalvorm

In Inleiding informatica hebben we gezien dat 'normaliseren' twee dingen kan inhouden: elimineren van herhalende groepen en elimineren van redundantie. In deze paragraaf kijken we naar het elimineren van een herhalende groep.

Eerste normaalvorm Genormaliseerd = 1NV

Een relationele structuur zonder herhalende groepen heet *genormaliseerd*. We kunnen ook zeggen: hij voldoet aan de *eerste normaalvorm* (1NV)). Die eerste normaalvorm zegt dus niets anders dan dat de structuur voldoet aan het verbod op herhalende groepen.

Ter illustratie kijken we naar een tabel Student met *drie* kolommen: naam, geslacht en cursussen. Zie figuur 3.1, waarin ter illustratie ook een kleine voorbeeldpopulatie is opgenomen. (Deze tabel heeft niets te maken met de OpenSchool-database.)

Student

naam	geslacht	cursussen				
		cursus	studiepunten	datum	inschrijvingstype	tarief
Jan	m	wiskunde	2	2012-08-25	E	300
		natuurkunde	3	2012-08-25	E	300
Eva	v	wiskunde	2	2012-08-26	T	150

FIGUUR 3.1 Tabel Student met herhalende groep

Toelichting
– De kolom naam is identificerend voor een student (niet echt realistisch, maar voor dit voorbeeld is dat geen probleem).
– De kolom cursussen is een herhalende groep van cursussen waarvoor een student is ingeschreven. Merk op dat de voorbeeldpopulatie slechts twee rijen heeft.
– Cursussen worden geïdentificeerd door een unieke naam.
– De inschrijvingstypen E en T duiden op 'examenstudent' respectievelijk 'toehoorder'.

De kolom naam zouden we kunnen benoemen tot primaire sleutel van tabel Student. Hoewel dit vanuit conceptueel oogpunt mogelijk is, is het gebruikelijk de begrippen primaire sleutel, verwijssleutel, enzovoort te reserveren voor een structuur die minimaal in 1NV staat.

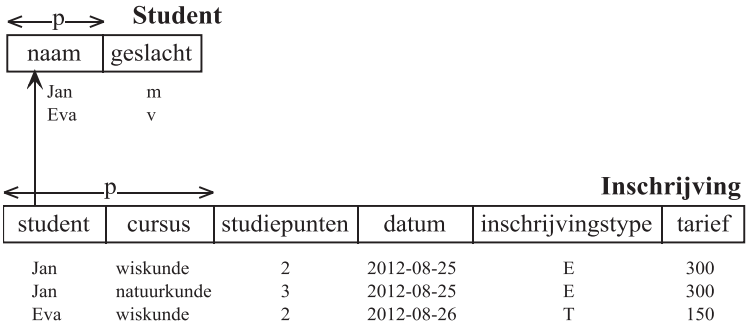
Om de structuur van figuur 3.1 te transformeren naar 1NV, gaan we net zo te werk als bij het Toetjesboek in Inleiding informatica. Hiertoe splitsen we de herhalende groep af.

Van oudertabel Student blijven alleen de identificerende kolom naam en de kolom geslacht over. De kolom naam wordt primaire sleutel.

Afgesplitste herhalende groep wordt *kind*tabel

Omdat de herhalende groep een meervoudig kenmerk ('cursussen') van een student bevat, wordt de afgesplitste tabel een kindtabel van Student. Als we onze tekenconventie om ouders zoveel mogelijk boven hun kinderen te tekenen in acht nemen, betekent dit dus dat de afgesplitste tabel *onder* het restant van de oorspronkelijke tabel komt. Dit is altijd zo bij het afsplitsen van een herhalende groep. Kort gezegd: *een afgesplitste herhalende groep gaat naar beneden*.

In het Student-voorbeeld bevat elke rij van de afgesplitste tabel één cursusinschrijving. Als tabelnaam kiezen we daarom Inschrijving. Omdat elke inschrijving bij één student hoort, moeten we een kolom met verwijzingen naar Student toevoegen. Als kolomnaam kiezen we: student.
Een inschrijving wordt geïdentificeerd door een combinatie van een student en een cursus. De kolomcombinatie (student, cursus) wordt dus de primaire sleutel. Zie figuur 3.2.



FIGUUR 3.2 Structuur in 1NV

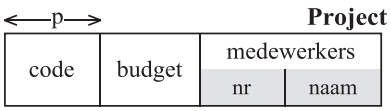
De nieuwe structuur staat in 1NV en is dus 'netjes relationeel'.

Naamgeving sleutelkolommen
We houden er in dit boek vrij strakke naamgevingsconventies op na. Dat schept eenheid en duidelijkheid en leidt ook tot strakke SQL-code.
De kolomnaam van een smalle primaire sleutel luidt meestal 'nr', 'code' of 'naam'. Een verwijssleutelkolom heeft meestal dezelfde naam als de corresponderende tabel, maar dan met een kleine letter.
Deze conventie voor verwijssleutels kan een misverstand opleveren: dat de verwijzing vanzelf ontstaat door de naamgeving. Dat is niet het geval! We moeten daar apart SQL-code voor schrijven, zoals we zullen zien wanneer we SQL/DDL behandelen.

OPGAVE 3.1

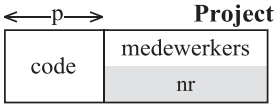
Figuur 3.3 bevat drie varianten, steeds met wat minder informatie, over de medewerking aan projecten door werknemers van een bedrijf. Iedere werknemer kan aan meerdere projecten werken. Normaliseer alle tabellen tot 1NV.

a



b

c



FIGUUR 3.3 Niet-genormaliseerde tabellen

De vraag is nu of de nieuwe structuren (in figuur 3.2 en de uitwerking van opgave 3.1) redundantie toelaten. Voor we deze vraag beantwoorden gaan we in op een fundamenteel begrip dat nauw in verband staat met redundantie: het begrip *functionele afhankelijkheid*. Aan de hand daarvan kunnen we meteen nog eens laten zien wat nu precies bedoeld wordt met de term 'redundant'.

2 Functionele afhankelijkheid

Functionele afhankelijkheid

Het begrip *functionele afhankelijkheid* is al meermaals impliciet aan de orde geweest. Eerst behandelen we, in paragraaf 2.1, een vorm van functionele afhankelijkheid zoals die in elke 'goede' tabel voorkomt. Maar het gaat om paragraaf 2.2: structuren die bij een gegeven functionele afhankelijkheid ongewenst zijn. Deze paragraaf biedt niet zozeer nieuwe inzichten, als wel een nieuwe taal voor dingen die al bekend zijn.

2.1 FUNCTIONELE AFHANKELIJKHEID EN SLEUTELS

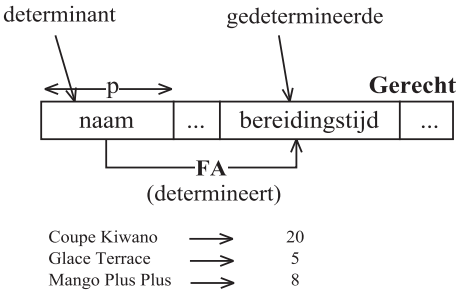
In een relationele database worden 'feiten' bewaard die betrekking hebben op het bestaan van bepaalde 'dingen' en op eigenschappen van die dingen. Zo wordt in het Toetjesboek in de tabel Gerecht het bestaan van de gerechten 'Coupe Kiwano', 'Glace Terrace' en 'Mango Plus Plus' uitgedrukt door de gegevens in de primaire-sleutelkolom naam. In combinatie met die naam drukken de andere kolommen eigenschappen van een gerecht uit, waaronder een bereidingstijd.

Functioneel afhankelijk

Determinant
Gedetermineerde

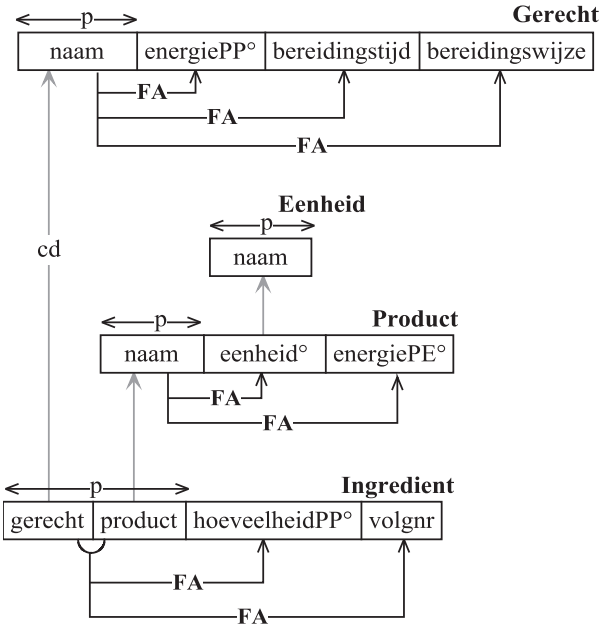
Pijl met FA

We zeggen: bereidingstijd is *functioneel afhankelijk* van naam. Dit betekent niets anders dan dat bij één waarde van naam één waarde van bereidingstijd hoort. De tabel Gerecht is er om die waarde (en andere gerechteigenschappen) te kunnen beheren en op te zoeken. We noemen naam de *determinant*, en bereidingstijd de *gedetermineerde*. We zeggen: naam *determineert* bereidingstijd. In een strokendiagram drukken we functionele afhankelijkheid uit door een pijl met de letters FA, zie figuur 3.4. Let op de richting van de pijl, en lees de letters FA eventueel als 'determineert'.



FIGUUR 3.4 Functionele afhankelijkheid: naam determineert bereidingstijd

Overal waar we een tabel hebben met een primaire sleutel en een of meer andere kolommen, zijn die andere kolommen functioneel afhankelijk van de primaire sleutel. Dat zijn ze ook van een alternatieve sleutel, die is immers evenals een primaire sleutel geschikt om 'dingen' te identificeren. Zie figuur 3.5 voor alle functionele afhankelijkheden binnen de tabellen van het Toetjesboek.



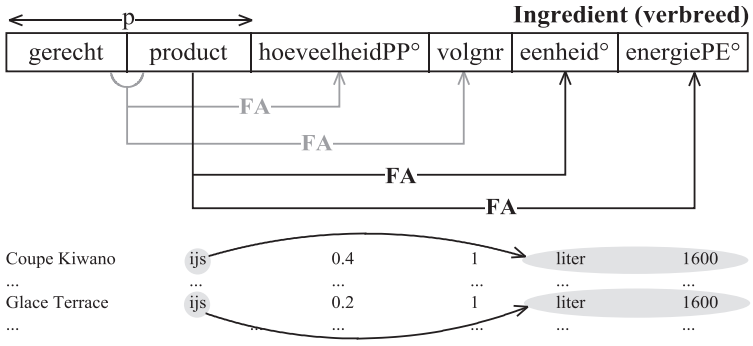
FIGUUR 3.5 'Gewone' functionele afhankelijkheden: in een genormaliseerde structuur

Merk op dat figuur 3.5 ten opzichte van het Toetjesboek uit Inleiding informatica een kleine toevoeging heeft: Ingredient heeft een extra kolom volgnr gekregen, die wordt gebruikt om de ingrediënten in de juiste volgorde af te drukken, zodat bijvoorbeeld niet 'peper' voorop komt.

2.2 FUNCTIONELE AFHANKELIJKHEID IN EEN ONGEWENSTE STRUCTUUR

We zien in Ingredient dat hoeveelheid$_{PP}$ en volgnr functioneel afhankelijk zijn van de (brede) primaire sleutel (gerecht, product). Via de verwijzing naar Product horen hier ook een eenheid en een energie$_{PE}$ bij. Deze zijn echter functioneel afhankelijk van alléén product, vandaar dat ze niet in de tabel Ingredient thuishoren. Zetten we ze er toch in, dan zijn we aan het *denormaliseren* en krijgen een structuur die redundantie toelaat: zie figuur 3.6.

Denormaliseren



FIGUUR 3.6 Functionele afhankelijkheden in een ongewenste (niet volledig genormaliseerde) structuur

De kern van de zaak is deze: figuur 3.6 bevat functionele afhankelijkheden waarvan de determinant niet uniek is. Omdat de determinant (product) vaker dan één keer kan voorkomen, kunnen ook de bijbehorende eenheid en de bijbehorende energie$_{PE}$ vaker dan één keer (dus *redundant*) vermeld worden. Dat kan problemen opleveren bij mutaties: als we op een gegeven moment besluiten om ijs voortaan in grammen te meten, moet dat op alle plekken waar nu 'ijs' aan 'liter' gekoppeld is worden aangepast. Bovendien bestaat bij redundante gegevensopslag het gevaar van *inconsistentie* (tegenstrijdigheid van gegevens), bijvoorbeeld als de eenheid van ijs de ene keer geschreven wordt als 'liter' en de andere keer als 'l'. Naast onduidelijkheid over de juiste schrijfwijze geeft dit onvolledige zoekacties, bijvoorbeeld bij zoeken op 'liter'.

Redundant

Inconsistentie

De 'hogere normaalvormen' die in deze en volgende paragrafen worden behandeld, hebben alle te maken met het uitbannen van redundantie. Voordat we echter naar die normaalvormen kunnen gaan kijken, moeten we ons bewust zijn van wat we wel en niet uit een populatie kunnen afleiden, en onder welke omstandigheden.

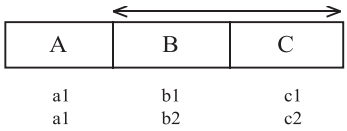
3 Populaties

Voorbeeldpopulatie

Illustratieve populatie

We komen in deze cursus twee soorten populaties tegen: voorbeeldpopulaties en illustratieve populaties. Een *voorbeeldpopulatie* is precies wat de naam al zegt: een mogelijke inhoud van een database. Die inhoud moet natuurlijk voldoen aan alle regels die in de database gelden. Een *illustratieve populatie* is ook een voorbeeldpopulatie, maar met een sterkere eigenschap: een illustratieve populatie illustreert zo goed mogelijk wat wel en niet mag, gegeven de regels van een database.

We bekijken een klein abstract voorbeeld om het verschil duidelijk te maken, zie figuur 3.7.



FIGUUR 3.7 Voorbeeldpopulatie of illustratieve populatie?

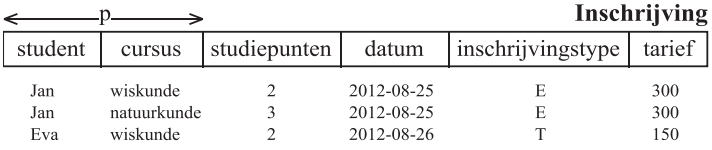
Volgens onze tekenconventie zijn alleen de strengste regels die gelden getekend, wat in dit geval wil zeggen dat er geen smalle uniciteitsregels gelden. De getoonde populatie is dus niet illustratief, want ze laat bijvoorbeeld niet zien dat kolom B niet uniek is. Een illustratieve populatie moet laten zien dat alle gegeven regels gelden én dat alle niet gegeven regels niet gelden (zie ook Inleiding informatica, leereenheid 12 'Relationele databases: regels').

De populatie in figuur 3.7 is dus een 'gewone' voorbeeldpopulatie: er is niets mis mee, maar er kan van alles in ontbreken. Anders gezegd: uit de *aan*wezigheid van tweemaal a1 in kolom A kunnen we afleiden dat kolom A niet uniek is, maar uit de *af*wezigheid van tweemaal dezelfde waarde in kolom B kunnen we niets afleiden.

Belangrijk

U moet er altijd van uitgaan dat een gegeven populatie niets meer of minder is dan een voorbeeldpopulatie. Alleen als we expliciet zeggen dat het om een illustratieve populatie gaat, mag u dingen afleiden uit de afwezigheid van gegevens.

We bespreken bovenstaande nogmaals aan de hand van een concreter voorbeeld, dat we vervolgens gebruiken om de tweede normaalvorm uit te leggen. We gaan uit van tabel Inschrijving van figuur 3.2. We herhalen deze in figuur 3.8.

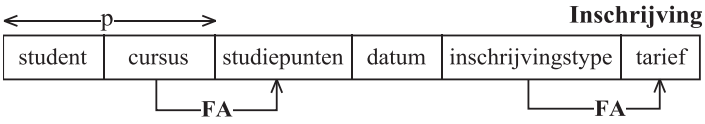


FIGUUR 3.8 Tabel Inschrijving met voorbeeldpopulatie: zijn er ongewenste functionele afhankelijkheden?

De voorbeeldpopulatie is erg klein maar roept toch een paar vragen op:
– Heeft de cursus wiskunde *altijd* 2 studiepunten, ongeacht wie zich ervoor inschrijft? In het algemeen: wordt het aantal studiepunten functioneel bepaald door de cursus?
– Wordt het tarief functioneel bepaald door het inschrijvingstype? Betaalt dus *elke* examenstudent 300 (euro) en *elke* toehoorder 150?

Zonder nadere informatie van een deskundige medewerker van de onderwijsinstelling zijn deze vragen niet te beantwoorden. De genoemde functionele verbanden zijn immers semantisch van aard, en de semantiek van gegevens kunnen we niet uit de gegevens zelf afleiden. Alleen als ons van tevoren was verzekerd dat de populatie illustratief is, hadden we deze vragen zonder de hulp van een deskundige kunnen beantwoorden.

We nemen nu aan dat we zo'n deskundige bij de hand hebben en dat deze zegt: 'Ja, die verbanden bestaan, het is geen kwestie van toeval in deze voorbeeldpopulatie'. Dit betekent dat er (naast de functionele afhankelijkheden met de sleutel als determinant) nog twee andere functionele afhankelijkheden bestaan, zie figuur 3.9.



FIGUUR 3.9 Twee functionele afhankelijkheden met niet-unieke determinant

De twee FA's die in figuur 3.9 zijn aangegeven staan op gespannen voet met de gegeven structuur, omdat hun determinant niet uniek is. Beide FA's zijn dus ongewenst, maar er is nog onderscheid te maken in de reden daarvoor. Dat doen we in de volgende twee paragrafen.

4 De tweede normaalvorm

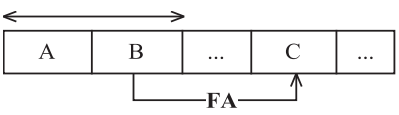
Tweede normaalvorm 2NV

De determinant cursus is niet uniek, omdat deze een *deel* is van een brede sleutel. Dit is precies de verbodsbepaling horende bij de *tweede normaalvorm* (2NV). De structuur van figuur 3.9 staat dus niet in 2NV. Haar hoogste normaalvorm is 1NV.

Partiële sleutelafhankelijkheid

Niet-sleutelkolom

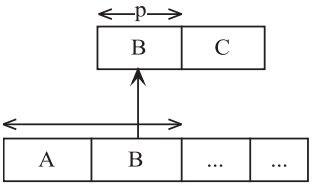
Figuur 3.10 geeft algemeen aan welke situatie wordt verboden door 2NV: een niet-sleutelkolom (of kolomcombinatie) C die functioneel afhankelijk is van een *deel* B van een brede sleutel AB. In andere woorden: 2NV verbiedt *partiële sleutelafhankelijkheid*. Houd hierbij in de gaten dat met 'sleutel' steeds een kandidaatsleutel wordt bedoeld. Een *niet-sleutelkolom* is dus een kolom die niet deel uitmaakt van een kandidaatsleutel.



FIGUUR 3.10 Situatie die wordt verboden door 2NV: partiële sleutelafhankelijkheid

Het recept om partiële sleutelafhankelijkheid kwijt te raken is de combinatie BC onder te brengen in een aparte tabel (zie Inleiding informatica). Dit wordt altijd een *ouder*tabel, omdat we daarin de informatie die nu mogelijk redundant in de oorspronkelijke tabel staat willen kunnen opzoeken. Kolom B wordt de primaire sleutel van de nieuwe oudertabel. Kolom C gaat ook mee en blijft functioneel afhankelijk van B, maar in de nieuwe tabel is dat zonder redundantie.

Kolom C (de gedetermineerde) verdwijnt uit de oorspronkelijke tabel. Kolom B (de determinant) blijft in de oude tabel gehandhaafd en wordt verwijzende sleutel naar de nieuwe tabel. Zie figuur 3.11.

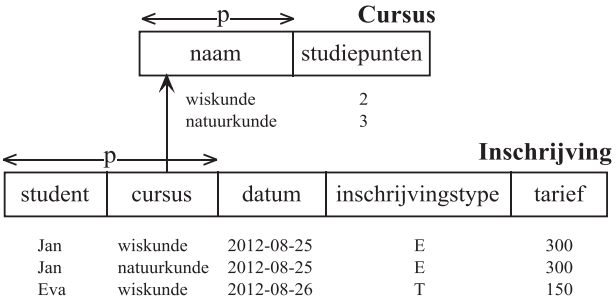


FIGUUR 3.11 Structuur in 2NV, na afsplitsing van de partieel sleutelafhankelijke combinatie

'Afgesplitste redundantie' wordt *ouder*tabel

Kort gezegd: *'afgesplitste redundantie' gaat omhoog*. Dit geldt ook voor de hogere normaalvormen die redundantie elimineren.

Passen we dit recept toe op tabel Inschrijving, dan krijgen we figuur 3.12. Merk op dat verwijzende sleutel (cursus) en primaire sleutel (naam) verschillende namen hebben. Dit is een gevolg van onze naamgevingsconventie. Beide kolommen bevatten cursusnamen, zoals de voorbeeldpopulatie illustreert.



FIGUUR 3.12 Structuur in 2NV, maar niet in 3NV

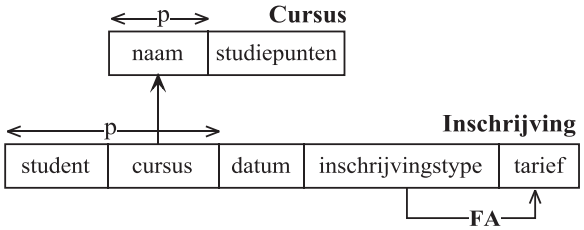
Zoals u ziet worden de combinaties (cursus, studiepunten) in de nieuwe tabel niet meer meervoudig opgenomen. In het bijzonder komt (wiskunde, 2) nog maar één keer voor; de redundantie is verdwenen.

OPGAVE 3.2

Welke van de uitwerkingen bij opgave 3.1 voldoen niet aan de tweede normaalvorm? Transformeer deze naar 2NV.

5 De derde normaalvorm

De structuur van figuur 3.12 bevat nog steeds een FA die in de gegeven structuur redundantie kan veroorzaken: een niet-sleutelkolom die functioneel afhankelijk is van een andere niet-sleutelkolom. Zie figuur 3.13 en ook de oorspronkelijke figuur 3.9.

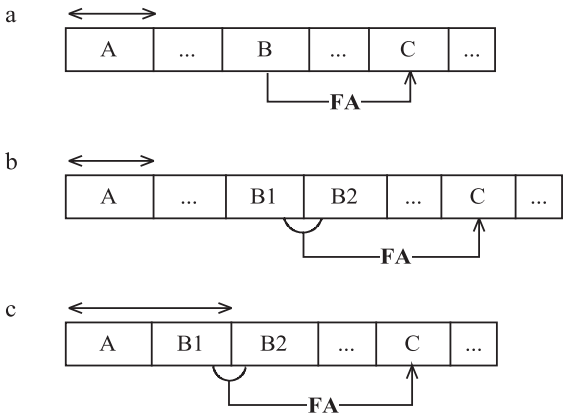


FIGUUR 3.13 Structuur in 2NV, maar niet in 3NV

Derde normaalvorm 3NV

Dit is een overtreding van de specifieke verbodsbepaling van de *derde normaalvorm* (3NV). De structuur van figuur 3.13 staat dus niet in 3NV. Haar hoogste normaalvorm is 2NV.

Figuur 3.14 illustreert de specifieke situatie die, in aanvulling op 2NV, wordt verboden door 3NV: een niet-sleutelkolom C die functioneel afhankelijk is van een niet-sleutelkolom of niet-sleutelkolomcombinatie B.



FIGUUR 3.14 Specifieke verbodsovertredingen van 3NV: transitieve sleutelafhankelijkheid

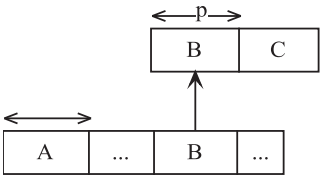
Opmerking: elke kolom of kolomcombinatie is functioneel afhankelijk van de sleutel A. Deze afhankelijkheden zijn als gewoonlijk niet getekend.

Transitieve sleutelafhankelijkheid

De verboden situatie heet *transitieve sleutelafhankelijkheid*. Met 'transitief' wordt bedoeld dat C *via* B van de sleutel A afhankelijk is. Dit komt bovenop de verbodseis van 2NV, het verbod op partiële sleutelafhankelijkheid.

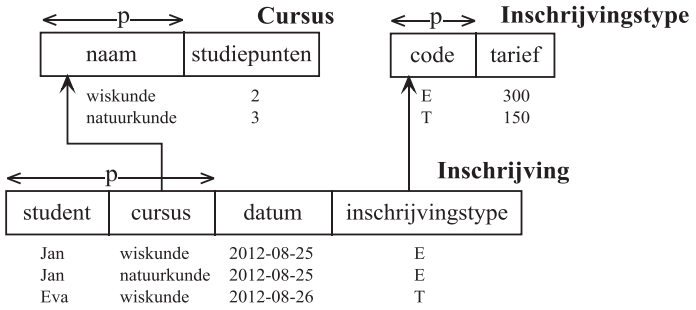
De figuur illustreert drie speciale gevallen van transitieve sleutelafhankelijkheid:
a B is één enkele niet-sleutelkolom
b B is een combinatie van twee niet-sleutelkolommen B1 en B2
c B is een combinatie van een sleutelkolom B1 en een niet-sleutelkolom B2.

Het recept om een transitieve sleutelafhankelijkheid kwijt te raken is hetzelfde als dat voor partiële sleutelafhankelijkheid: schrap de gedetermineerde (C) en breng deze samen met de determinant B onder in een aparte oudertabel. Figuur 3.15 brengt dit in beeld voor figuur 3.14a. Voor de figuren b en c gaat dit net zo (maar dan met samengestelde sleutels B1, B2).



FIGUUR 3.15 Structuur in 3NV, na afsplitsing van de transitief sleutelafhankelijke combinatie

Passen we dit toe op figuur 3.13, dan is het resultaat figuur 3.16. De structuur hiervan voldoet aan 3NV. Alle overgebleven functionele afhankelijkheden (welke zijn dat?) zijn nu *gewenste functionele afhankelijkheden*.



FIGUUR 3.16 Structuur in 3NV

Afwijkende definities van 2NV en 3NV
De oorspronkelijke definities van 2NV en 3NV, door Codd, waren gesteld in termen van de *primaire sleutel*. Dat was onbevredigend, omdat het daarmee, bij meerdere kandidaatsleutels, van de keuze van de primaire sleutel kon afhangen wat de hoogste normaalvorm is. In deze tekst houden we ons aan de meer recente definities, die zijn gesteld in termen van 'sleutel', dat wil steeds zeggen: kandidaatsleutel. Ook met dit uitgangspunt echter komt u in de literatuur subtiele verschillen en niet-equivalente definities tegen. U bent dus gewaarschuwd.

OPGAVE 3.3

Figuur 3.17 bevat een niet-genormaliseerde relationele structuur, met voorbeeldpopulatie, die betrekking heeft op bibliotheekboeken. Elk van de zes rijen bevat informatie over één boek. Elk boek heeft een uniek boeknummer (primaire sleutel), een eveneens uniek maar niet verplicht isbn, een titel, de code en de naam van de uitgever (beide identificerend voor de uitgever) en auteursinformatie. Van elke auteur is een auteurnummer gegeven (identificerend) en de naam van de auteur (niet identificerend).

p

Boek

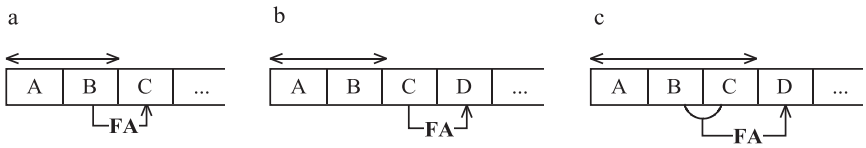
nr	isbn°	titel	uitgevercode	uitgevernaam	auteurs	
					nr	naam
101	90-6277-160-0	Sprookjes	BM	Boom	37	Bomans
103	90-2754-233-1	Databases	SH	Stonehenge	50	Andersen
					77	Date
105	90-6277-277-0	Eric	BM	Boom	37	Bomans
110		Flipperen	SU	Surprise	87	Jacobse
115	90-415-207-0	Sprookjes	BM	Boom	63	Andersen
129	90-6277-1348	Relaties	SH	Stonehenge	77	Date
					56	Murre
					93	Blanken

FIGUUR 3.17 Tabel Boek (niet-genormaliseerd)

Normaliseer de structuur in stappen (1NV, 2NV, 3NV). Kies in elk stadium duidelijke tabelnamen en kolomnamen. Teken steeds alle sleutels en uniciteitsregels, en de functionele afhankelijkheden die in combinatie met de structuur redundantie kunnen veroorzaken.

OPGAVE 3.4

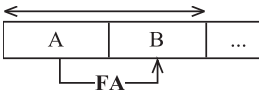
De structuren van figuur 3.18 bevatten elk een functionele afhankelijkheid die in de gegeven structuur ongewenst is, vanwege een niet-unieke determinant. Normaliseer elke structuur tot 3NV.



FIGUUR 3.18

OPGAVE 3.5

In figuur is 3.19 de uniciteitspijl in combinatie met de functionele afhankelijkheid in strijd met de tekenconventie voor uniciteitsregels. Waarom?



FIGUUR 3.19

6 De Boyce-Codd-normaalvorm

Hogere normaalvormen

We hebben gezien dat de specifieke verbodsbepaling op 3NV transitieve sleutelafhankelijkheid is. Omdat 3NV tevens 2NV veronderstelt, wordt door 3NV ook partiële sleutelafhankelijkheid verboden. De vraag is nu of we met 3NV alle vormen van ongewenste functionele afhankelijkheid hebben geëlimineerd.

Het antwoord hierop is niet zo eenvoudig. 3NV blijkt nog een vorm van functionele afhankelijkheid toe te laten die – vanwege een niet-unieke determinant – aanleiding kan geven tot redundantie. Er is dus ruimte voor een nog hogere normaalvorm, die ook deze vorm van functionele afhankelijkheid verbiedt. Dit is de Boyce-Codd-normaalvorm. We zullen echter zien dat het onwenselijk kan zijn deze bijzondere vorm van functionele afhankelijkheid te elimineren.

6.1 DE VERBODSBEPALING VAN BCNV

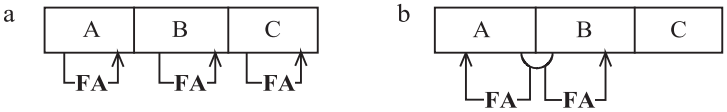
Boyce-Codd-normaalvorm BCNV

De verbodsbepaling van de *Boyce-Codd-normaalvorm* (BCNV) is iets algemener dan die van 3NV en verbiedt dus net wat meer:

BCNV verbiedt:
1 herhalende groepen (de verbodsbepaling van 1NV)
2 *niet-triviale* functionele afhankelijkheden met niet-unieke determinant.

Triviale en niet-triviale FA's

De toevoeging 'niet-triviaal' vraagt om uitleg bij het begrip 'triviale FA'. Elke kolom of kolomcombinatie determineert zichzelf, en elke kolomcombinatie determineert zijn eigen kolommen, zie figuur 3.20.



FIGUUR 3.20 Triviale functionele afhankelijkheden

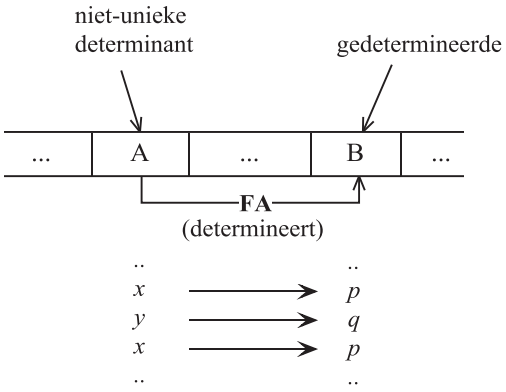
Immers:
– figuur a: bij een gegeven waarde *x* van A hoort altijd één waarde van A (namelijk *x* zelf)
– figuur b: bij een gegeven waardecombinatie (*x*, *y*) van A en B hoort altijd één waarde van A (namelijk *x* zelf) en van B (namelijk *y* zelf).

Triviale functionele afhankelijkheid

Dit soort functionele afhankelijkheden worden *triviaal* genoemd, ze vallen natuurlijk niet onder de verbodsbepaling.

Structuren zonder herhalende groep en zonder 'niet-triviale FA's met niet-unieke determinant' voldoen dus per definitie aan de Boyce-Codd-normaalvorm.

In figuur 3.21 wordt de tweede voorwaarde nog eens geïllustreerd. De situatie die daar schematisch wordt aangegeven, is precies de situatie die door BCNV wordt verboden: een kolom B die functioneel afhankelijk is van een *niet-unieke* determinerende kolom A. Waarden die in kolom A meervoudig voorkomen (in de figuur: de waarde *x*), hebben vanwege de FA gelijke B-waarden (in de figuur: *p*). Schrappen we in het voorbeeld een van de twee *p*'s, dan kunnen we deze weer reconstrueren uit de context: redundantie.



FIGUUR 3.21 Typerende situatie die wordt verboden door BCNV

Dit hele verhaal gaat ook op wanneer de determinant samengesteld is (A is dan een combinatie van twee of meer kolommen).

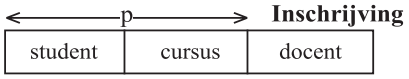
6.2 EEN KRITISCH VOORBEELD VOOR BCNV

Kritisch voorbeeld BCNV

Ter illustratie van wat BCNV meer verbiedt dan 3NV, kijken we naar een *kritisch voorbeeld* voor BCNV: een structuur die wel in 3NV staat, maar niet in BCNV.

VOORBEELD 3.1

Figuur 3.22 geeft een nieuwe variant op de tabel Inschrijving, met een voorbeeldpopulatie. Ook hier wordt een inschrijving geïdentificeerd door een combinatie (student, cursus). De derde kolom docent geeft een kenmerk van zo'n inschrijving: de student schrijft zich voor een cursus in bij een bepaalde docent.

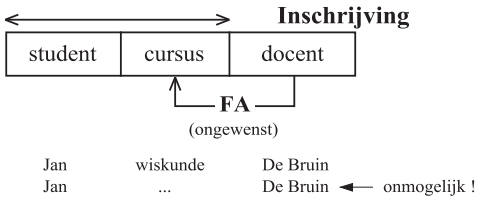


Jan	wiskunde	De Bruin
Jan	natuurkunde	De Bruin
Michiel	wiskunde	De Bruin
Michiel	natuurkunde	De Wit

FIGUUR 3.22 Inschrijving: student kan per cursus maar één docent hebben

We voegen nu nog een extra aanname toe: dat elke docent maar één vak geeft. Deze aanname komt neer op een functionele afhankelijkheid: een docent (de determinant) determineert een vak (de gedetermineerde).

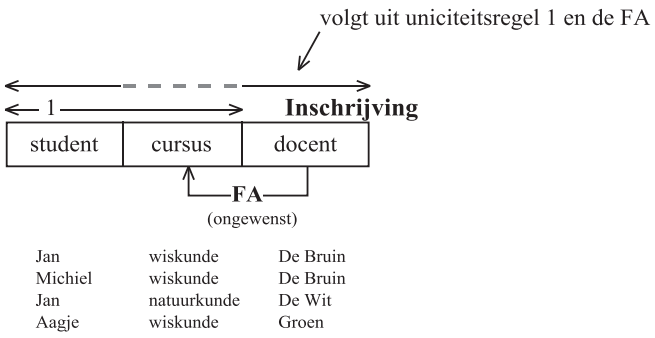
Zie figuur 3.23.



FIGUUR 3.23 Kritisch voorbeeld BCNV

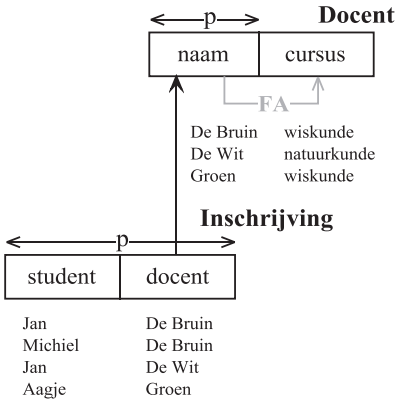
Er is hier géén sprake van partiële sleutelafhankelijkheid en evenmin van transitieve sleutelafhankelijkheid. De structuur staat dus in 3NV. Toch heeft de FA een niet-unieke determinant, zodat de structuur niet aan BCNV voldoet. We hebben hier dus een kritisch voorbeeld voor BCNV te pakken.

De populatie van figuur 3.22 voldoet niet meer; immers De Bruin geeft daar twee vakken. Laten we aannemen dat Jan zich voor wiskunde heeft ingeschreven bij De Bruin, en dat De Bruin kennelijk wiskunde geeft. Merk nu op dat de combinatie (Jan, De Bruin) geen tweede keer kan voorkomen: dat zou immers alléén met wiskunde kunnen, maar de combinatie (Jan, wiskunde) is uniek en bestaat al, zie weer figuur 3.23. We mogen dus concluderen dat de gegeven uniciteitsregel en de functionele afhankelijkheid samen nog een tweede uniciteitsregel impliceren, zie figuur 3.24.



FIGUUR 3.24 Samenhang uniciteitregels en functionele afhankelijkheid

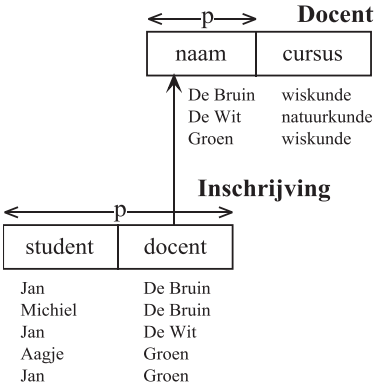
Omdat de determinant van de FA (kolom docent) niet uniek is, laat deze structuur redundantie toe. De populatie van figuur 3.24 illustreert dit: zouden we 'wiskunde' in de eerste of in de tweede rij schrappen, dan konden we deze waarde reconstrueren. De FA lijkt dus ongewenst en we zouden hem moeten elimineren, zoals we ook bij de kritische voorbeelden voor 2NV en 3NV hebben gedaan. Het resultaat zien we in figuur 3.25 en voldoet aan BCNV. Determinant en gedetermineerde vormen in de nieuwe structuur een aparte tabel (Docent) en de gedetermineerde is uit de oorspronkelijke tabel (Inschrijving) geschrapt. De FA (grijs getekend) is er natuurlijk nog steeds, maar hoeft niet te worden getekend omdat de determinant nu een kandidaatsleutel is.



FIGUUR 3.25 Kritisch voorbeeld BCNV, na normalisatie

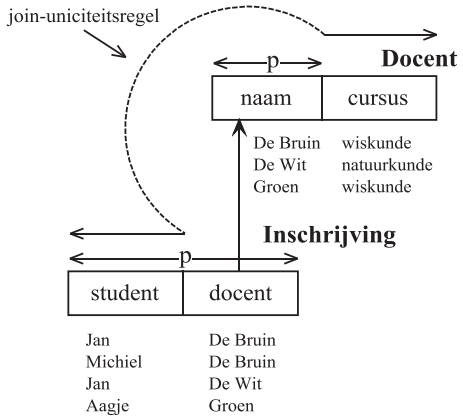
Wie nu denkt dat de structuur van figuur 3.25 gelijkwaardig is met die van figuur 3.11, heeft het mis. Het verhaal gaat nog verder.

Een join-uniciteitsregel
Het opmerkelijke van dit voorbeeld is dat we door te normaliseren van 3NV naar BCNV de oorspronkelijke uniciteitsregel, over (student, cursus), zijn kwijtgeraakt. Figuur 3.26 illustreert dat de combinatie (Jan, wiskunde) in de nieuwe structuur best twee keer kan voorkomen, met twee verschillende docenten.



FIGUUR 3.26 In deze structuur is de student-cursuscombinatie niet uniek

De nieuwe structuur is pas gelijkwaardig aan de oude als we de oorspronkelijke uniciteitsregel weer toevoegen. Hij loopt nu echter over twee kolommen in verschillende tabellen, via de verwijzende sleutel die de docentnamen koppelt, zie figuur 3.27.



FIGUUR 3.27 De verloren uniciteitsregel: teruggevonden als join-uniciteitsregel

Join-uniciteitsregel

De uniciteitsregel heet in de nieuwe structuur een *join-uniciteitsregel*. Die naam komt van een essentiële operatie op tabellen: de join, dat is het samenvoegen van twee tabellen tot één tabel, door de bij elkaar horende rijen te zoeken. Als we in figuur 3.27 achter iedere Inschrijving-rij de bijbehorende docentinformatie plakken (dat is in dit geval alleen de cursus die door die docent gegeven wordt), dan krijgen we precies de tabel Inschrijving van figuur 3.24. En dat is dus de join van Inschrijving en Docent uit figuur 3.27. In leereenheid 5 gaan we uitgebreid in op de join-operator.

Merites van het kritische voorbeeld

We kunnen ons afvragen wat we zijn opgeschoten door de 'ongewenste FA' te elimineren en hiervoor een join-uniciteitsregel voor in de plaats te krijgen. Is die join-uniciteitsregel niet minstens zo onwenselijk? In paragraaf 7.2 komen we op deze kwestie terug.

7 Kanttekeningen

We beëindigen onze bespreking van de normalisatietheorie met enige kanttekeningen.

7.1 1NV VERSUS DE HOGERE NORMAALVORMEN

In Inleiding informatica hebben we vastgesteld dat herhalende groepen en redundantie los staan van elkaar: we kunnen het een hebben zonder het ander en omgekeerd. In wezen hebben de hogere normaalvormen (2NV, 3NV en BCNV) dan ook weinig te maken met de eerste normaalvorm (1NV). In de definities van de hogere normaalvormen zit echter ook de eerste normaalvorm besloten. Deze verstrengeling heeft een heldere theorievorming niet direct bevorderd. Tegenwoordig beseft men dat een rigoureus verbod op herhalende groepen niet vanzelfsprekend is. Zeker nu moderne relationele databases het gebruik van bepaalde gestructureerde datatypen toestaan, is het verbod aan heroverweging toe. Want wat is een herhalende groep anders dan een kolom met een specifiek gestructureerd datatype? In deze leereenheid bewandelen we echter de klassieke weg, vanuit de eerste normaalvorm.

7.2 DE (ON)WENSELIJKHEID VAN REDUNDANTIE

Impliciet hebben we tot nu toe de boodschap overgebracht dat redundantie 'verkeerd' is. En dus ook dat normaliseren 'goed' is. Meestal klopt dit ook wel, maar toch niet altijd.

Redundante structuren ter wille van performance
Redundantie wordt soms bewust voor lief genomen, ter wille van de performance. Dat is met name het geval wanneer we hierdoor vermijden dat tijdrovende joins gemaakt moeten worden. Om geen inconsistente database-inhoud te krijgen moeten er dan ofwel geen updates zijn (zoals in *datawarehouses*, dat zijn grote historiedatabases) ofwel moet er een bewakingsmechanisme zijn dat inconsistentie uitsluit.

De merites van BCNV
In voorbeeld 3.1 hebben we gezien dat door normalisatie een functionele afhankelijkheid werd geëlimineerd, maar dat we er een join-uniciteits-regel voor in de plaats kregen.

Denkend vanuit de oorspronkelijke visie op inschrijvingen (als combinaties van een student en een vak), valt toch wel te betreuren dat door te normaliseren die combinaties hun 'eigen tabel' zijn kwijtgeraakt. In plaats daarvan wordt ons een andere visie op inschrijvingen opgedrongen: als combinaties van een student en een docent. Is die tweede visie even goed? We kunnen heel goed verdedigen dat dit *niet* het geval is, omdat een gebruiker een student-cursuscombinatie als het wezen (of zelfs als de definitie) van een inschrijving ziet, en diezelfde gebruiker de functionele afhankelijkheid van figuur 3.23 uitsluitend ziet als een extra beperkende regel die los staat van de betekenis van een inschrijving. Voor zo'n gebruiker is het zelfs heel goed denkbaar dat die regel wordt losgelaten, zonder dat daardoor de visie op wat een inschrijving hoeft te worden herzien.

Zachte en harde semantiek

Wat gebeurt nu bij het normaliseren? Daar wordt niet gekeken naar deze zogenaamde 'zachte' semantiek, maar uitsluitend naar de 'harde' semantiek van uniciteitsregels en functionele afhankelijkheden. De beide uniciteitsregels die vanuit de 'zachte' semantiek nogal verschillend van karakter zijn, worden in de harde semantiek gelijkwaardig. Daardoor kan de tweede uniciteitsregel een sleutel worden in de nieuwe structuur, en de eerste (die nog wel direct verbonden is met wat de gebruiker zich voorstelt bij een inschrijving) de nogal technische vorm aannemen van een join-uniciteitsregel.

BCNV niet per se beter dan 3NV.

Conclusie: de 3NV-structuur van figuur 3.23 (inclusief de FA-beperkingsregel) valt te prefereren boven de BCNV-structuur van figuur 3.27 (met daarin de join-uniciteitsregel als extra beperkingsregel).

Normaliseren: alleen intratabelcriteria
Het voorgaande drukt ons met de neus op een fundamentele beperking van het normalisatieproces: alle normalisatiecriteria zijn geformuleerd in termen van *intra*tabelcriteria (criteria die gaan over zaken *binnen* tabellen). Anders gezegd: een database voldoet aan de *n*-de normaalvorm (*n* = 1, 2, 3, BC) als elke tabel afzonderlijk in de *n*-de normaalvorm staat.

Voorbeeld 3.1 leert ons dat normaliseren soms met zich meebrengt dat ongewenste *intra*tabelkenmerken worden geëlimineerd ten koste van nieuwe ongewenste *inter*tabelkenmerken. De nuchtere conclusie luidt: ook bij databases moeten we niet alles willen.

7.3 DE VIERDE EN DE VIJFDE NORMAALVORM

Met de Boyce-Codd-normaalvorm is het verhaal van de klassieke normalisatietheorie nog niet afgelopen. De theorie kent nog twee hogere normaalvormen: de vierde normaalvorm (4NV) en de vijfde normaalvorm (5NV). Deze staan voor nog strengere verbodsbepalingen met betrekking tot structuren die redundante gegevensopslag mogelijk maken. Het gaat hierbij om redundantie die voortkomt uit vormen van afhankelijkheid die algemener zijn dan functionele afhankelijkheid. De kritische voorbeelden (wel BCNV/niet 4NV, wel 4NV/niet 5NV) hebben een nogal academisch karakter en zijn voor de praktijk nauwelijks interessant.

In de praktijk is het zo dat een structuur die in 3NV staat, bijna altijd ook in BCNV én in 4NV én in 5NV staat en dus 'volledig genormaliseerd' is.

Normalisatietheorie als controlemiddel
Wie in één keer een goede structuur opzet, met 'elk soort ding zijn eigen tabel', hoeft niet meer te normaliseren of te standaardiseren. In de praktijk is de normalisatietheorie vooral van belang als controlemiddel en voor correctie van structuren die ondanks alle goede bedoelingen toch nog redundantie bevatten. Verder geeft de theorie ons een taal om over kenmerken van structuren te communiceren.
Omgekeerd hoeft in een genormaliseerde structuur niet elk 'soort ding' zijn eigen tabel te hebben. Normalisatie dwingt immers niet altijd standaardisatie af, waar dat wel gewenst is.

SAMENVATTING

Paragraaf 1

Normaliseren houdt in: elimineren van herhalende groepen en elimineren van redundantie. Een structuur zonder herhalende groepen staat per definitie in de eerste normaalvorm (1NV) en heet *genormaliseerd*.

Paragraaf 2

Een kolom B heet *functioneel afhankelijk* van een kolom of kolomcombinatie A in dezelfde tabel wanneer de regel geldt dat bij elke A-waarde steeds dezelfde B-waarde hoort. We zeggen: A *determineert* B. A heet de *determinant*, B de *gedetermineerde*. We zijn alleen geïnteresseerd in niet-triviale gevallen, dat wil zeggen dat B niet gelijk is aan A of een deelverzameling is van A.

Wanneer A een kandidaatsleutel is, is elke B functioneel afhankelijk van A. Dit is het normale en 'gewenste' geval van functionele afhankelijkheid.

Wanneer één bepaalde waarde A daadwerkelijk vaker voorkomt, zijn de meervoudige vermeldingen van de corresponderende B-waarden redundant. Deze (meestal ongewenste) situatie kan zich voordoen wanneer voor A geen uniciteitsregel geldt.

Paragraaf 3

Er is een verschil tussen voorbeeldpopulaties en illustratieve (of representatieve) populaties. Een *voorbeeldpopulatie* is niets meer of minder dan een verzameling voorbeeldrijen bij een databaseschema. Deze populatie moet voldoen aan alle gegeven regels, maar er kunnen geen regels uit afgeleid worden.
Een *illustratieve populatie* laat naast de geldende regels ook zien dat alle niet gegeven regels niet gelden.

Paragraaf 4

De *tweede normaalvorm* (*2NV*) eist 1NV en verbiedt *partiële sleutelafhankelijkheid*: een niet-sleutelkolom die functioneel afhankelijk is van een echt deel van een kandidaatsleutel.

Een structuur die 2NV overtreedt kan redundante gegevens bevatten. De oplossing is: een structuurwijziging, waarbij de combinaties van determinant (A) en gedetermineerde (B) in een eigen tabel worden geplaatst waarvan A de primaire sleutel wordt en waarin eenmalig ('single point of definition') bij elke A-waarde de B-waarde wordt vermeld. De B-kolom verdwijnt uit de oorspronkelijke tabel, de A-waarden daarin worden verwijzingen naar de nieuwe tabel.

Paragraaf 5

De *derde normaalvorm* (*3NV*) eist 2NV en verbiedt *transitieve sleutelafhankelijkheid*: een niet-sleutelkolom die indirect (via een andere niet-sleutelkolom of -combinatie) functioneel afhankelijk is van een kandidaatsleutel.
Ook een structuur die 3NV overtreedt kan redundante gegevens bevatten. De oplossing is: een structuurwijziging analoog aan die voor 2NV.

Paragraaf 6

De *Boyce-Codd-normaalvorm* (BCNV) is nog wat sterker dan 3NV. Een tabel voldoet aan BCNV als hij voldoet aan 1NV en geen functionele afhankelijkheid bevat met een niet-unieke determinant.

Een kritisch voorbeeld (3NV en niet BCNV) leert ons dat het uitbannen van een *intra*tabelregel (een redundantie veroorzakende functionele afhankelijkheid) een *inter*tabelregel kan introduceren (een joinuniciteitsregel). De vraag is gerechtvaardigd wat we daarmee zijn opgeschoten. BCNV is niet beter dan 3NV.

Paragraaf 7

De 1NV-eis (geen herhalende groepen) voor de hogere normaalvormen is theoretisch gezien niet vanzelfsprekend.
Structuren die redundantie toestaan (geen 3NV) kunnen soms nuttig zijn, met name als de performance erdoor verbeterd wordt.

ZELFTOETS

Bij alle opgaven wordt verondersteld dat u in elk stadium alle tabellen een geschikte naam geeft en dat u sleutels, uniciteitsregels en verwijzingen tekent.

1 Bepaal de hoogste normaalvorm (kies uit 1NV t/m BCNV) waaraan PQRS voldoet (zie figuur 3.28).

PQRS

P	Q	R	S
p1	q1	r1	s1
p2	q1	r1	s2
p1	q2	r2	s3
p3	q1	r1	s1
p3	q2	r2	s4
p4	q3	r3	s2
p5	q3	r3	s3

FIGUUR 3.28

2 Op een conferentie worden de presentaties beoordeeld door de deelnemers. De deelnemers vullen hiertoe voor elke presentatie een formulier in, waarbij ze een cijfer mogen geven voor verschillende aspecten van de presentatie. Iedere presentatie heeft één spreker. In figuur 3.29 zijn enkele resultaten verzameld.

Formulier

deelnemer	presentatie	titel	sprekernr	sprekernaam	beoordelingen		
					code	omschrijving	cijfer
A1	1	Use cases	1	Theo	V	vorm	7
					I	inhoud	8
A1	2	Missing values	5	Rie	V	vorm	6
					I	inhoud	9
A2	1	Use cases	1	Theo	V	vorm	7
					I	inhoud	7

FIGUUR 3.29

In elke rij zijn gegevens over één formulier opgenomen:
– een unieke deelnemerscode (de beoordeling is niet anoniem)
– nummer en titel van de presentatie (beide uniek)
– nummer (uniek) en naam van de spreker van de presentatie
– in de vorm van een herhalende groep: code en omschrijving (beide uniek) van de aspecten waarop wordt beoordeeld en het daarvoor gegeven cijfer.

a Normaliseer tot 1NV (door alleen de herhalende groep af te splitsen).
b Zijn er functionele afhankelijkheden die bij de gegeven structuur redundantie kunnen veroorzaken? Zo ja, teken deze.
c Normaliseer de structuur verder tot BCNV.
d Voldoet de genormaliseerde structuur aan het principe 'elk entiteittype zijn eigen tabel'? Zo niet, pas de structuur aan.

3 Figuur 3.30 bevat een tabelstructuur die is ontworpen om informatie op te slaan over muzieknummers op muziekdragers (cd, dubbel-cd, driedubbel-cd, lp, dubbel-lp, enzovoort). Elke rij bevat gegevens over één muziekdrager.

Muziekdrager

nr	naam	type	aantal_kanten	groep	groepsleden	nummers			
					naam	kant	track	titel	tijd

FIGUUR 3.30

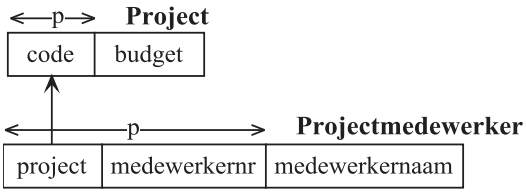
Toelichting
– Elke muziekdrager heeft een identificerend nummer, een naam en een type (bijvoorbeeld cd of lp). De kolomwaarde aantal_kanten bevat het aantal kanten: bijvoorbeeld 2 voor een gewone lp, 4 voor een dubbel-lp en 2 voor een dubbel-cd.
– Er geldt de aanname dat elke muziekdrager door één muziekgroep is opgenomen. Elke groep heeft een identificerende naam en bestaat uit een of meer groepsleden, onafhankelijk van de muziekdrager. De kolom 'groepsleden' vormt een herhalende groep. De waarden ervan zijn subtabellen met één kolom, genaamd 'naam'. De namen identificeren de groepsleden. Iemand kan lid zijn van meer dan één groep.
– Elke muziekdrager bevat een of meer muzieknummers. Elk nummer wordt uniek bepaald door de combinatie van het volgnummer van de 'kant' en het tracknummer óp die kant. Onder kant verstaan we hier zowel elke zijde van een lp, als ook elke afzonderlijke cd.
– Voorbeeld van een kant- en trackaanduiding: 2, 8. Dit kan track 8 zijn op kant 2 van een lp, maar ook track 8 op de tweede cd van een dubbel-cd.
– Elk nummer heeft één titel en duurt één bepaalde tijd in seconden.

Opdracht
Transformeer de gegeven structuur stapsgewijs tot 3NV. Ga daarbij iets anders te werk dan in de vorige opgave.
a Uit de toelichting volgt een functionele afhankelijkheid waarvan de determinant niet uniek is en de gedetermineerde een herhalende groep. Elimineer als eerste stap deze functionele afhankelijkheid en laat daarbij de herhalende groep intact.
b Elimineer nu beide herhalende groepen (normalisatie tot 1NV).
c Teken functionele afhankelijkheden die ongewenst zijn in combinatie met de structuur die in b is verkregen.
d Normaliseer tot 3NV.

TERUGKOPPELING

1 Uitwerking van de opgaven

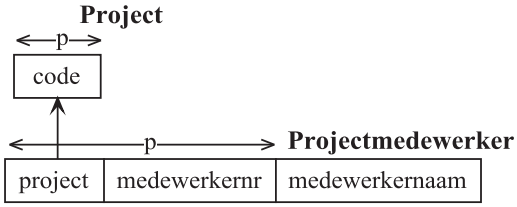
3.1 a Bij afsplitsing van een herhalende groep wordt de primaire sleutel van de oorspronkelijke tabel 'meegenomen' in de vorm van een verwijssleutel. Het resultaat is een 'kind' met een verwijzing naar een 'ouder'. Vaak (niet altijd!) wordt in de kindtabel de verwijssleutel onderdeel van een brede primaire sleutel. Hier is dat inderdaad het geval (zie figuur 3.31).



FIGUUR 3.31

De tabelnaam Projectmedewerker is gekozen omdat deze tabel projecten koppelt aan medewerkers. De kolomnamen nr en naam zijn aangepast, om ook in de nieuwe context duidelijk te maken dat het kenmerken zijn van een werknemer.

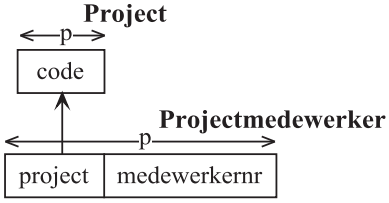
b Er verandert niets wezenlijks (zie figuur 3.32).



FIGUUR 3.32

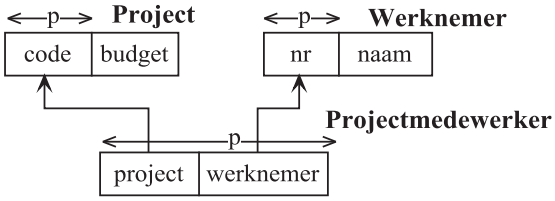
Vanuit onderdeel a gezien is dit vrij logisch, toch blijkt dit randgeval vaak lastig gevonden te worden.

c Dit onderdeel illustreert een herhalende groep met één kolom. Dit is nog meer een randgeval dan onderdeel b. Maar ook nu verandert er niets wezenlijks (zie figuur 3.33).



FIGUUR 3.33

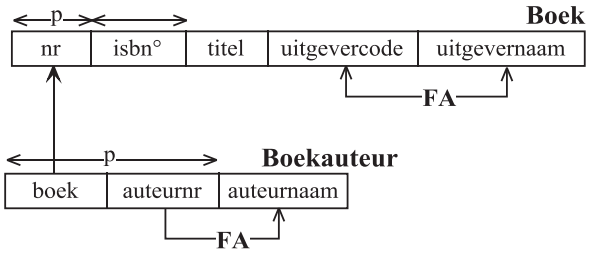
3.2 In de figuren 3.31 (onderdeel a) en 3.32 (onderdeel b) bevat tabel Projectmedewerker een partiële sleutelafhankelijkheid: medewerkernr → medewerkernaam. De structuren voldoen dus niet aan 2NV. In figuur 3.34 tonen we de genormaliseerde structuur voor onderdeel a. (Voor onderdeel b gaat het analoog.)



FIGUUR 3.34

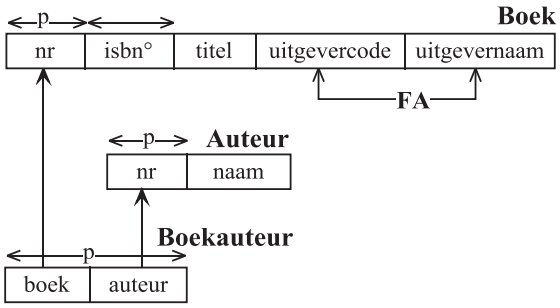
We hebben meteen de gelegenheid te baat genomen om na te denken over geschikte tabel- en kolomnamen: in opgave 3.1 ging het alleen over medewerkers aan projecten, en hebben we de term 'medewerker' steeds gewoon overgenomen uit de oorspronkelijke tabel. Nu we echter een aparte tabel maken waarin niet alleen projectmedewerkers, maar álle werknemers van het bedrijf kunnen worden opgenomen, kiezen we ervoor die nieuwe tabel Werknemer te noemen in plaats van Medewerker. De naam van de verwijzende kolom in Projectmedewerker veranderen we meteen mee, maar de naam van de tabel Projectmedewerker lijkt ons prima de lading dekken zo.

3.3 Na afsplitsen van de herhalende groep staat de structuur in 1NV (zie figuur 3.35).



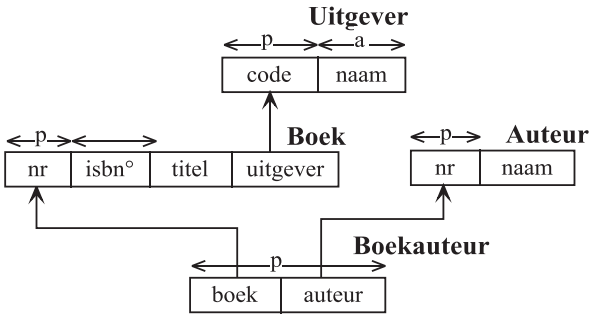
FIGUUR 3.35

De afgesplitste tabel heeft de naam Boekauteur gekregen. Elke rij bevat immers auteurinformatie in relatie tot een bepaald boek. Auteurregel was ook een goede naam geweest.
Merk op dat uitgevercode en uitgevernaam *wederzijds functioneel afhankelijk* zijn. In plaats van met twee aparte FA-symbolen is dit weergegeven met één FA-symbool met twee pijlpunten.
De functionele afhankelijkheid binnen Boekauteur impliceert een partiële sleutelafhankelijkheid. De structuur staat daardoor niet in 2NV. Na decompositie (afsplitsing van tabel Auteur) is dit wel het geval (zie figuur 3.36).



FIGUUR 3.36

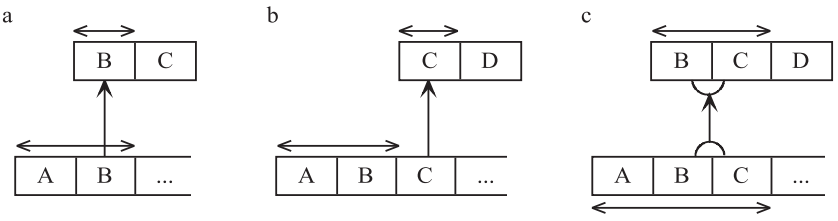
De twee getekende functionele afhankelijkheden binnen tabel Boek zijn van het type 'transitieve sleutelafhankelijkheid', zodat Boek, en daarmee de hele structuur, niet in 3NV staat.
Er zijn nu twee manieren van afsplitsen mogelijk, afhankelijk van wat we als primaire sleutel willen in de afgesplitste tabel: uitgevercode of uitgevernaam. Omdat bij de keuze van een primaire sleutel korte codes de voorkeur verdienen boven een langere naam, valt de keuze op uitgevercode (zie figuur 3.37).



FIGUUR 3.37

Merk op dat we de kolomnamen 'auteurnaam' en 'uitgevernaam' hebben veranderd in 'naam'. In de context van de tabellen Auteur en Uitgever zijn nadere kwalificaties immers niet nodig.

3.4 Zie figuur 3.38.



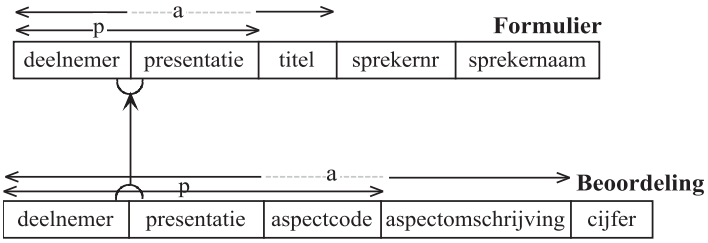
FIGUUR 3.38

3.5 Bij een gegeven waarde a voor kolom A hoort volgens de functionele afhankelijkheid precies één waarde b voor kolom B. Zou nu een waarde a twee keer voorkomen, dan zou volgens die functionele afhankelijkheid de combinatie (a, b) ook twee keer voorkomen, maar dat is in strijd met de gegeven uniciteitsregel over A en B. Geen enkele waarde a mag dus tweemaal voorkomen, en dat betekent dat er een uniciteitsregel over kolom A geldt. Dit is echter in strijd met de conventie om alleen de meest strenge uniciteitsregels als uniciteitspijl te noteren.

2 Uitwerking van de zelftoets

1 Indien, naast de uniciteitsregel, geen enkele beperkingsregel is gegeven, voldoet de structuur aan BCNV. (Dit was een strikvraag, om nog eens onder de aandacht te brengen dat uit een populatie nooit regels zijn af te leiden; hooguit kan worden afgeleid dat een bepaalde regel *niet* geldt.)

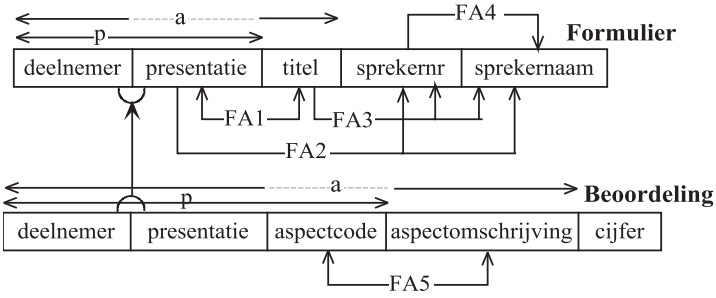
2 a Afsplitsen van de herhalende groep geeft de structuur in 1NV van figuur 3.39. Hierbij is de naamgeving verbeterd: de nieuwe kolomnamen aspectcode en aspectomschrijving geven expliciet aan om wat voor code en omschrijving het gaat. Het zijn immers geen codes en omschrijvingen van beoordelingen maar van aspecten waarop beoordeeld wordt.



FIGUUR 3.39

Ook hebben we in figuur 3.39 aangegeven dat de tabel Formulier behalve de gegeven primaire sleutel volgens de beschrijving ook een alternatieve sleutel heeft.

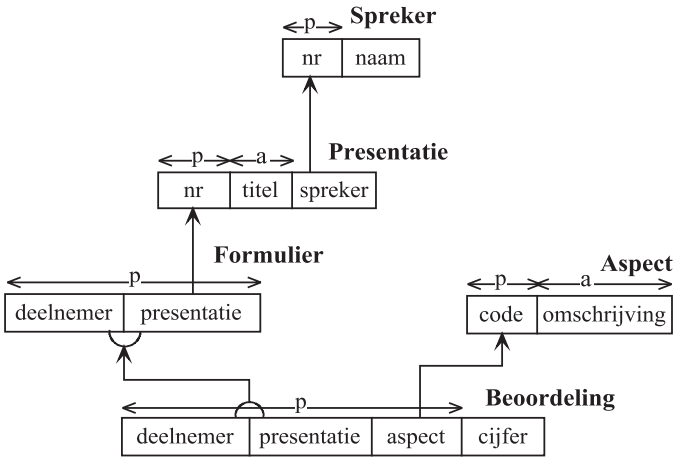
b De structuur van figuur 3.39 bevat vijf ongewenste functionele afhankelijkheden die tot redundantie aanleiding kunnen geven. Zie figuur 3.40. We hebben de twee wederzijdse functionele afhankelijkheden niet dubbel geteld.



FIGUUR 3.40

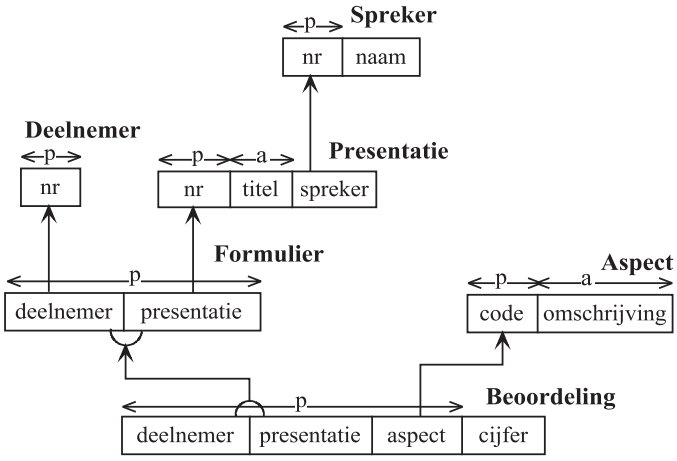
FA2 en FA3 zijn partiële sleutelafhankelijkheden, FA4 is een transitieve sleutelafhankelijkheid, en FA1 en FA5 zondigen specifiek tegen BCNV.

c Afsplitsen van alle FA's geeft een resultaat dat voldoet aan BCNV:



FIGUUR 3.41

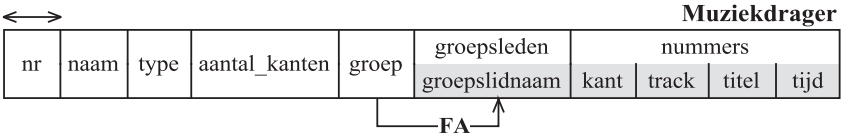
d De 'wereld' van de conferentie bevat nog een entiteittype dat geen eigen tabel heeft: het entiteittype 'deelnemer'. De deelnemernummers zijn hierdoor niet gestandaardiseerd. Ze worden 'gewoon' gebruikt in tabel Formulier. Al is het deelnemernummer het enige hier gebruikte kenmerk van deelnemers, dit is geen reden om zo'n eigen tabel niet te maken. Figuur 3.42 bevat de verbeterde structuur. Uiteraard kunnen andere kenmerken aan de tabel Deelnemer worden toegevoegd, zoals naam, e-mail, enzovoort.



FIGUUR 3.42

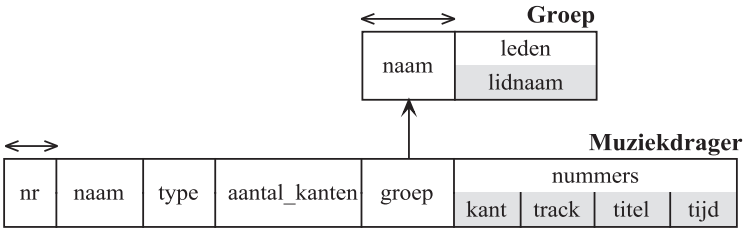
Let wel: deze stap maakt geen deel uit van het normalisatieproces. Mede hierom is normalisatie geen volwaardige modelleermethode.

3 a Figuur 3.43 toont de gewraakte functionele afhankelijkheid.



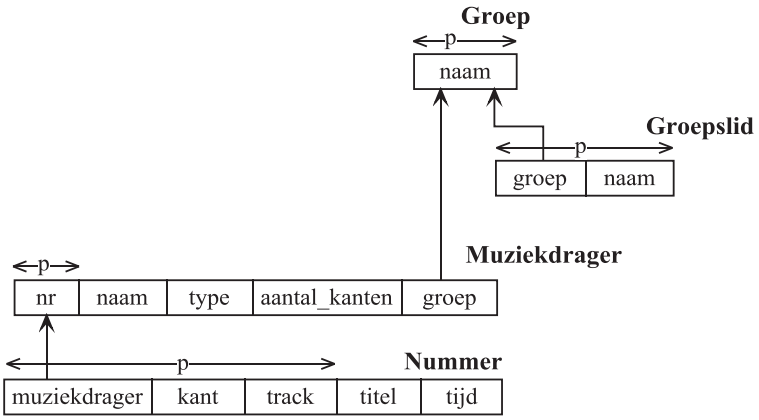
FIGUUR 3.43

Afsplitsen resulteert in de structuur van figuur 3.44.



FIGUUR 3.44

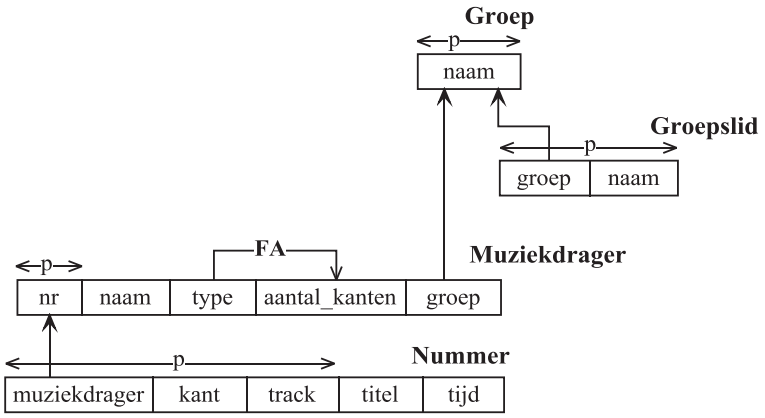
b Normalisatie tot 1NV leidt tot figuur 3.45. Zowel Groep als Muziekdrager krijgt er een kindtabel bij voor de meervoudige kenmerken uit hun respectievelijke herhalende groepen.



FIGUUR 3.45

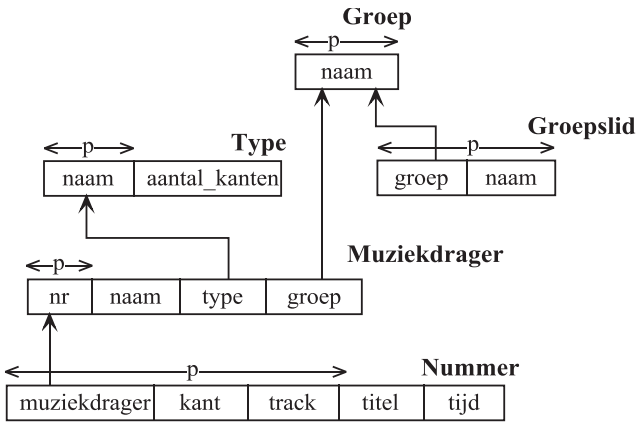
Merk op dat groepsnamen tengevolge van deze operatie zijn gestandaardiseerd in de 'eigen' tabel Groep, waarbij de groepsnamen in de tabellen Groepslid en Muziekdrager zijn onderworpen aan de referentiële-integriteitsregel.

c Zie figuur 3.46 voor de enige ongewenste functionele afhankelijkheid.



FIGUUR 3.46

d Het elimineren van de ongewenste afhankelijkheid leidt tot het strokendiagram in figuur 3.47. De structuur staat dan in 3NV (en ook in BCNV). Bijkomend voordeel: de type-aanduiding is nu ook gestandaardiseerd.



FIGUUR 3.47