The Third Manifesto: Setting the Record Straight (6 en slot)

Semantische compositionaliteit

H. Darwen en C.J. Date

Om hun stelling te onderbouwen dat operators als relaties kunnen worden behandeld duiken de auteurs van The Third Manifesto diep in de relationele algebra. Om het artikel leesbaar te houden beperken wij ons in deze publicatie tot de rode draad van hun redenering; op de website vindt u het volledige betoog.

Deel 3 van Gittens' artikel [2] heet "Het niet vasthouden aan de principes van semantische compositionaliteit". We citeren het hier in zijn geheel.

Het is een algemeen principe van taalontwerp dat substitutie van variabelen door hun corresponderende waarden de betekenis van de expressies waarin ze voorkomen niet mag veranderen. In de derde editie van The Third Manifesto (TTM) doen Date en Darwen dit principe geweld aan. Bedenk dat het type van een relvar wordt bepaald door de header van de relvar. De candidate keys horend bij een relvar zijn volgens TTM géén onderdeel van hun type. Deze keuze van Date en Darwen vormt een ernstige logische fout omdat het ervoor zorgt dat variabelen en waarden van hetzelfde type niet uitwisselbaar zijn. Dit blijkt duidelijk als u bedenkt dat een relatiewaarde C van type T niet toegewezen kan worden aan een relationele variabele V van type T. Om precies te zijn, de toewijzing van C aan V is niet toegestaan als er beperkingen voor candidate keys zijn gedefinieerd voor V waarvan C niet afhankelijk is. Anders gezegd: zelfs al zijn V en C van hetzelfde type, de toewijzing V = C kan wel of niet worden toegestaan, afhankelijk of C wel of niet afhankelijk is van alle constraints voor candidate keys die voor V gelden.

Datzelfde geldt voor geneste relatiewaarden en niet-geneste relatiewaarden, die nooit de rol van parent kunnen spelen in foreign key relaties, als we de faciliteiten die TTM biedt gebruiken. Dat dit overduidelijk waar is, blijkt uit het feit dat volgens TTM relatiewaarden geen bijbehorende candidate keys hebben en foreign keys worden uitgedrukt in termen van candidate keys van de parent relatievariabele. Ergo, volgens TTM kunnen relvars in het algemeen nooit worden vervangen door hun waarden en zijn als gevolg daarvan niet referentieel transparant. Ten slotte handelen Date en Darwen ook in strijd met het principe van conceptuele integriteit en hun eigen RM Prescriptie nummer 21. Dit blijkt uit het feit dat toewijzing van waarde v aan een variabele V, genoteerd als V : = v, waarbij V en v van hetzelfde type zijn, niet impliceert dat de gelijkheidsexpressie V = v in het algemeen waar is. Nogmaals, de reden voor deze inconsistentie is gelegen in het feit dat het type van de relatievariabelen niet de bijbehorende candidate keys bevat.

In DB/M 2 2007 uitte Maurice Gittens nogal wat kritiek op de derde editie van het standaardwerk 'Databases, Types, and the Relational Model: The Third Manifesto' van Hugh Darwen en Chris Date. Gittens vraagt zich af wat de bijdrage van Darwen en Date aan het relationele model de afgelopen jaren is geweest – de derde editie van TTM acht hij zelfs een regressie ten opzichte van de ideeën en principes van wijlen Ted Codd, grondlegger van het relationele model. Hij onderbouwt dit met zes argumenten. Het genoemde artikel 'Twijfels over logische correctheid' is een afgeleide van Gittens' publicatie op zijn website www.gittens.nl, zie [2].
H. Darwen en C.J. Date hebben Gittens' commentaar uiterst serieus genomen en krijgen van DB/M de gelegenheid hun standpunten en meningen ten aanzien van Gittens' opmerkingen diepgaand toe te lichten. Leerzame stof over de basisregels van het relationele database-model. Dit is de zesde en laatste bijdrage in een serie.

Tegenwoordig gebruiken we meestal de term key

Wij vonden sommige stukken uit deze tekst moeilijk te volgen, maar wat we ervan begrijpen bestrijden we krachtig. Door de genoemde moeilijkheid hebben we besloten de tekst puntsgewijs in stukken te knippen om zo ons standpunt uit te leggen. We beginnen met de titel van de paragraaf.

1. Het niet vasthouden aan de principes van semantische compositionaliteit.
Uit Wikipedia [3]: "The Principle of Compositionality[1] is the principle that the meaning of a complex expression is deter-

mined by the meanings of its constituent expressions and the rules used to combine them." Verderop staat in hetzelfde artikel "This principle is sometimes called Frege's Principle, because Frege is widely credited for the first modern formulation of it. However, the idea appears already among Indian philosophers of grammar such as Yāska, and also in Plato's work such as in *Theaetetus*." Dit wordt gevolgd door een nieuwe paragraaf: "The Principle of Compositionality also exists in a similar form in the compositionality of programming languages."
Het verband tussen de titel van Gittens' paragraaf en hetgeen daarop volgt, is ons niet helemaal duidelijk. We nemen aan dat elke acceptabele weerlegging van wat volgt dat tegelijk ook doet voor de titel.

2. Het is een algemeen principe van taalontwerp dat substitutie van variabelen door hun corresponderende waarden[2] de betekenis van de expressies waarin ze voorkomen niet mag veranderen.
We nemen aan dat "ze" refereert aan variabelen en we herinneren de lezer eraan dat in deze context de term "variabelen" verwijst naar het programmeertaal-concept met die naam en niet het wiskundige.
Hier is een expressie die variabele referenties bevat: x - y. De expressie beschrijft de subtractie van y van x. Verder kunnen we er niets meer uit lezen omdat de actuele waarden horend bij x en y van tijd tot tijd variëren. Neem nu eens aan dat op een bepaald moment de "corresponderende waarden" van x en y respectievelijk 2 en 1 zijn. Als we die waarden substitueren voor x en y, krijgen we de expressie: 2 - 1.

In plaats van een expressie die de subtractie van y van x beschrijft, hebben we er nu een die de subtractie van 1 van 2 beschrijft (en dus ook het nummer 1 beschrijft in de betekenis van "subtractie"). Het blijkt dus dat substitutie inderdaad "de betekenis van de expressie verandert." Daarom begrijpen we absoluut niet wat Gittens bedoelt met zijn openingszin. Dat heeft echter geen invloed op de daaropvolgende tekst, dus is het niet echt belangrijk.
Het woord 'subtractie', dat verwijst naar de operator die de variabelen in dit simpele voorbeeld met elkaar verbindt, verandert natuurlijk niet door de substitutie net zomin als de betekenis ervan.

3. In de derde editie van TTM doen Date en Darwen dit principe geweld aan.
Zoals we hebben uitgelegd begrijpen we niet welk principe, zo daar al sprake van is, we geweld aan zouden doen. Maar we kunnen met een gerust hart verklaren dat ook in TTM 'x - y' ook staat voor de subtractie van y van x en dat '2 - 1' ook de subtractie van 1 van 2 beschrijft. Dat geldt ook voor alle expressies die verwijzingen naar variabelen bevatten, inclusief relatievariabelen. Een voorbeeld: 'r1 UNION r2' beschrijft het samengaan van r1 en r2. Als we de relaties RELATION { TUPLE { X 1 } } en RELATION { TUPLE { X 2 } } substitueren, krijgen we de volgende expressie:

```
RELATION { TUPLE { X 1 } } UNION RELATION
                                { TUPLE { X 2 } }
```

Die expressie beschrijft het samenvoegen van RELATION { TUPLE { X 1 } } en RELATION { TUPLE { X 2 } }. Het woord UNION verandert niet als gevolg van de substitutie, haar betekenis evenmin.

4. Bedenk dat het type van een relvar wordt bepaald door de header van de relvar.
Dat klopt, behalve dan dat de juiste term *heading* is en geen *header*. Voor de duidelijkheid, en om redenen die buiten het kader van dit artikel vallen, gebruiken we normaliter de term *declared type* voor het type van een variabele. Als de heading van een relvar { H } is, waarbij H een lijst van attribuutdefinities is, dan is het declared type van de variabele RELATION { H }, wat inhoudt dat alleen waarden van type RELATION { H } kunnen worden toegekend aan aan die variabele.
We noemen de variabele een relatievariabele, of relvar, juist omdat zijn declared type een relatietype is.

5. De candidate keys horend bij een relvar zijn géén onderdeel van hun type volgens TTM.
Tegenwoordig gebruiken we meestal de term *key* in plaats van het vroegere *candidate key*. Een key van relvar r is een subset van de heading van r. In zekere zin is het een 'deel van' het declared type van r, omdat het deel uitmaakt van de heading, die op zijn beurt deel is van het declared type.

> Sommige database constraints verwijzen naar meer dan één variabele

Maar Gittens bedoelt vrijwel zeker niet de sleutels op zich, maar de impliciete beperkingen door declaraties van keys.
Het is juist om te zeggen dat de verzameling relaties die het declared type van r vormt een superset[3] is, mogelijk een echte superset van de verzameling relaties van dat type die voldoet aan al zijn beperkingen, inclusief vooral zijn key constraints. Met andere woorden, het is vrijwel zeker dat er relaties van het declared type van r zijn die *niet* voldoen aan de key constraints. Pogingen om zo'n relatie toe te wijzen aan r lopen stuk door *constraint violation* in *run time*; pogingen om een relatie toe te wijzen die niet tot het declared type van r behoort, mislukken in *compile time*.

6. Deze keuze van Date en Darwen vormt een ernstige logische fout omdat het ervoor zorgt dat variabelen en waarden van hetzelfde type niet uitwisselbaar zijn.
We begrijpen niet helemaal wat Gittens bedoelt met het uitwisselen van een waarde en een variabele. Eerder (bij punt 2, dat we ook al niet helemaal begrepen) refereert hij aan het vervangen van een variabele door zijn actuele waarde. Als twee dingen worden uitgewisseld neemt elk de plaats van de ander in – elk wordt gesubstitueerd door de ander.

7. Dit blijkt duidelijk als u bedenkt dat een relatiewaarde C van type T niet toegewezen kan worden aan een relationele variabele V van type T.
Terwijl toewijzing de ene waarde door een andere vervangt, is dit klaarblijkelijk geen substitutie van een waarde voor een variabele (of andersom). Voor alle duidelijkheid, de toewijzing is een syntactisch toegestane expressie maar die mislukt tijdens run time zoals dat het geval is bij *elke* beperking waaraan niet wordt voldaan. We kunnen begrijpen waarom een key constraint (onterecht) zou kunnen worden beschouwd als deel van de type definitie, maar database constraints in het algemeen kunnen onmogelijk zo worden gezien: want sommige database constraints verwijzen naar meer dan één variabele (zoals bijvoorbeeld meestal bij key constraints het geval is). Inderdaad, sommige relationele toewijzingen lukken soms wel en soms niet, afhankelijk van de inhoud van andere variabelen in de database.

8. Om precies te zijn, de toewijzing van C aan V is niet toegestaan als er beperkingen voor candidate keys zijn gedefinieerd voor V waaraan C niet gehecht is.
Nogmaals: zo'n toewijzing is een toegestane expressie die tijdens run time zal mislukken. Ook kunnen andere constraints veroorzaken dat het in run time zal mislukken. En het is geen logische fout (zie punt 6) om toe te zien op het naleven van de beperkingen. Het is wel een logische fout om dat *niet* te doen.

> Het is geen logische fout om toe te zien op het naleven van beperkingen

9. Anders gezegd: zelfs al zijn V en C van hetzelfde type, de toewijzing V = C kan wel of niet worden toegestaan, afhankelijk of C wel of niet gehecht is aan alle constraints voor candidate keys die voor V gelden.
Of de toewijzing slaagt hangt niet helemaal (zoals Gittens suggereert) af van de gehechtheid aan key constraints. De toewijzing kan ook in strijd zijn met andere beperkingen dan key constraints.

*10. Datzelfde geldt voor geneste relatie*waarden *en niet-geneste relatie*waarden*, die nooit de rol van parent kunnen spelen in foreign key relaties, als we de faciliteiten die TTM biedt gebruiken.*
TTM biedt geen "faciliteiten". De auteurs stellen een verzameling regels voor waaraan een relationele database-taal zou moeten voldoen.
We begrijpen ook niet waarom het woord 'waarden' zo wordt benadrukt. Een foreign key bevat twee relatie*variabelen*, meestal verschillende. Deze twee worden meestal aangeduid als de refererende relvar (waarbij de foreign key hoort) en de referentie relvar (waarbij de referentie key hoort). De referentie key wordt soms aangeduid als de parent, analoog aan de terminologie van het hiërarchische datamodel, maar we vinden die term niet goed passen en gebruiken hem niet. TTM eist van een taal dat deze de expressie van elke beperking ondersteunt, die kan worden uitgedrukt in een Tutorial D expressie in de vorm van IS_EMPTY (r), waarbij r een relationele expressie is van arbitraire complexiteit. De beperking, die impliciet is aan de foreign key declaratie (in relvar r1)

```
FOREIGN KEY ( a ) REFERENCES r2
```

is kort voor de Tutorial D expressie

```
IS_EMPTY ( r1 { a } NOT MATCHING r2 { a } )
```

Maar toevallig bevat Tutorial D op dit moment geen korte expressie voor een foreign key[4], noch vereist TTM dat. Zoals het bovengenoemde equivalent laat zien eist TTM dat beperking in een expressie kan worden aangeduid, zelfs als a een attribuut heeft waarvan het declared type een relatietype is (we maken dit punt voor het geval het relevant is voor Gittens' verwijzingen naar 'geneste' en 'niet-geneste' relaties, waar we niets van begrijpen). Punt 10 lijkt te zeggen dat bepaalde beperkingen niet zouden kunnen worden uitgedrukt in een bestaande taal. In dat geval zijn we snel klaar met punt 10: het zit er volledig naast.

11. Dat dit overduidelijk waar is, blijkt uit het feit dat volgens TTM relatiewaarden geen bijbehorende candidate keys hebben en foreign keys worden uitgedrukt in termen van candidate keys van de parent relatievariabele.
We begrijpen niet waar Gittens naar toe wil. Net zoals we niets begrijpen van punt 10, weten we absoluut niet waar Gittens op doelt als hij claimt dat iets "overduidelijk waar is". Daarnaast merken we op dat je van een relatie die voldoet aan een key constraint zou kunnen zeggen dat deze de key 'bezit' die de beperking impliceert, maar de term 'key' is normaliter gereserveerd voor relvars – hoe dan ook zou de term *superkey* beter op zijn plaats zijn (bedenk bijvoorbeeld, dat een relatie die niet meer dan één tuple bevat voldoet aan elke key constraint die voor een relvar van zijn type is gedefinieerd, maar zijn enige

echte key is de lege verzameling – en meestal hebben relvars waaraan kan worden toegewezen geen lege keys).

12. Ergo, volgens TTM kunnen relvars in het algemeen nooit worden vervangen door hun waarden en zijn als gevolg daarvan niet referentieel transparant.
We hebben de term 'referentieel transparant' moeten opzoeken. Wikipedia meldt: "An expression is said to be referentially transparent if it can be replaced with its value without changing the program (in other words, yielding a program that has the same effects and output on the same input)." Deze definitie is een beetje losjes, een waarde is immers geen onderdeel van programmeertekst, maar we snappen de bedoeling. Referentiële transparantie (wat we er nu van weten is hoofdzakelijk iets dat we gewoon maar aannemen) vereist duidelijk niet dat er naar elke waarde van het declared type van een variabele te allen tijde op een toegestane manier verwezen wordt.

Types en integriteits-beperkingen zijn logisch gescheiden concepten

13. Tenslotte handelen Date en Darwen ook in strijd met het principe van conceptuele integriteit en hun eigen RM Prescriptie nummer 21.
Conceptuele integriteit bereik je door trouw te blijven aan de concepten waarvoor je gekozen hebt. We hopen dat we dat bereikt hebben. We zijn opgelucht dat we in staat zijn geweest Gittens' pogingen om het tegenovergestelde aan te tonen, te weerleggen. Zoals aangegeven in het volgende punt, vereist RM Prescriptie 21 dat direct na de toewijzing van waarde v aan variabele V, de vergelijking v = V resulteert in WAAR.

14. Dit blijkt uit het feit dat toewijzing van waarde v aan een variabele V, genoteerd als V : = v, waarbij V en v van hetzelfde type T zijn, niet impliceert dat de gelijkheidsexpressie V = v in het algemeen waar is.
Integendeel; Gittens heeft ons niets laten zien in TTM dat in strijd is met RM Prescriptie 21. Op dit punt gekomen, willen we het volgende verzoeken aan iedereen die aspecten van een computertaalontwerp of een bepaalde computertaal wil bekritiseren, er vragen over stellen of bediscussiëren: als dat van toepassing is, illustreer dat dan alstublieft met voorbeelden, bij voorkeur gebruik makend van correcte syntax (we hebben Tutorial D tenslotte ontwikkeld om *onze* ideeën te kunnen illustreren). Hoewel we geloven dat we Gittens' kritiek hebben weerlegd, hebben we eerlijk toegegeven dat we niet overal begrijpen welk punt hij nu eigenlijk wil maken. Voor zover we weten heeft hij uiteindelijk één valide punt naar voren gebracht dat ons aan het denken heeft gezet. We willen hem dan ook van harte uitnodigen om ons daar opnieuw materiaal over te sturen, maar dan voorzien van voorbeelden die de door hem waargenomen inconsistentie aantonen – zo zou bijvoorbeeld een voorbeeld bij punt 10, waar we niets van begrijpen, bepaald nuttig kunnen zijn!

15. Nogmaals, de reden voor deze inconsistentie is gelegen in het feit dat het type van de relatievariabelen niet de bijbehorende candidate keys bevat.
En als het dat wél zou doen, welke toewijzingen V : = v zouden dan resulteren in WAAR die niet dat resultaat opleveren in TTM? *We rest our case.*

We besluiten met onze lezers eraan te herinneren, dat het doel van een *type* is om de operators te bepalen die beschikbaar zijn voor waarden en variabelen van dat type; beperkingen, *constraints*, dienen om de logische consistentie te onderhouden van de verzameling intergerelateerde variabelen die een database vormen. Types en integriteitsbeperkingen zijn logisch gescheiden concepten en elke poging om daarmee te knoeien moet krachtig worden bestreden.

Noten

1. *[3] bevat het woord 'semantisch' niet, maar het is uit andere teksten die we op het web gevonden hebben duidelijk dat dit het principe is waaraan Gittens refereert.*
2. *We geloven dat Gittens hier eigenlijk substituties van waarden voor variabelen bedoelt, zoals punt 12 verderop lijkt aan te tonen. We borduren voort op die aanname.*
3. *Eigenlijk een echte superset, behalve in het geval de volledige heading de enige key van r vormt.*
4. *Deze zaak zijn we al enige tijd aan het bestuderen maar de oplossing is moeilijk. De constructie die wordt gevonden in bijvoorbeeld SQL gaat gebukt onder veel te veel restricties.*

Literatuur

1. *C.J. Date and Hugh Darwen: Databases, Types, and the Relational Model: The Third Manifesto (3rd edition). Boston, Mass.: Addison-Wesley (2006).*
2. *Maurice Gittens: The Third Manifesto Revisited www.gittens.nl/TheTTMRevisited.pdf.*
3. *Diverse auteurs: Wikipedia. http://en.wikipedia.org.*

Hugh Darwen en **Chris Date** zijn auteurs van de derde editie van The Third Manifesto.

Dit is een bewerkte en vertaalde versie van de originele Engelse tekst, die u kunt vinden op onze website www.dbm.nl onder 'specials', 'extra materiaal'. In geval van discussies is de originele Engelse tekst doorslaggevend. In vorige artikelen is het woord 'compositie' gebruikt als vertaling van 'compositionality'. We geven in dit artikel er de voorkeur aan 'compositionaliteit' te gebruiken.