Theoretisch raamwerk achter de essentiële dataconstructie

Regel Nul

Frido van Orden

Een systeem dat zich wil kwalificeren als een RELATIONEEL, DATABASE, MANAGEMENT systeem moet zijn RELATIONELE faciliteiten (en niets anders) gebruiken om de DATABASE te MANAGEN.

Dit is het eerste echte artikel in de serie over Codd's regels voor relationele database management-systemen en het is meteen een vreemde eend in de bijt. In de meeste publicaties wordt gerefereerd aan 'de 12 regels', daarmee doelende op de regels 1 t/m 12 zoals die in het inleidende artikel van deze serie zijn geformuleerd. Er is echter ook een 'Regel 0' die feitelijk los staat van deze 12 regels maar zeker geen plaats als appendix verdient. Door de speciale nummering komt deze regel niet alleen bovenaan in de lijst, maar springt hij bovendien direct in het oog. En dat is nu precies waar het om te doen is.
Op het eerste gezicht trapt regel 0 een open deur in: een relationeel database management systeem is een systeem dat een database managet met behulp van relationele faciliteiten. De crux schuilt hem in het kleine bijzinnetje 'en niets anders'. Wat bedoelt Codd hier precies?

Een soort keurmerk

Het idee zou kunnen ontstaan dat de deur hier dichtgegooid wordt voor pre-relationele producten met een relationeel laagje eroverheen. Een van de belangrijkste redenen voor Codd om zijn regels te publiceren, was immers om een soort keurmerk voor relationele databases te creëren en daarmee het gras voor de voeten weg te maaien van pre-relationele producten die zich na enkele cosmetische aanpassingen onterecht als 'relationeel' aanprezen. Toch is dat niet de kern van het verhaal. Codd's afkeer van de relationele aspiraties van systemen als Cullinet IDBMS/R berustte niet op het feit dat deze systemen in de kern niet relationeel waren. Immers, een van de kernwaarden van het relationele model is het volstrekt scheiden van model en implementatie.
De ware relationeel kan zich dus nooit storen aan het feit dat een relationeel product (of een product dat claimt relationeel te zijn) intern niet vanuit een relationele visie is opgezet. Dat zijn implementatie-aspecten die niet relevant zijn.
Anders wordt dit wanneer deze implementatie-aspecten door de relationele laag heen zichtbaar worden. Dit kan bijvoorbeeld door het introduceren van niet-relationele uitbreidingen. Een goed voorbeeld hiervan is het gebruik van pointers in SQL, wat inmiddels zelfs in de officiële SQL standaard is doorgedrongen (REF-gegevenstypen). Er zijn dan ook vele artikelen te wijden aan de redenen waarom SQL geen goede relationele taal is, en Chris Date (Codd zelf overigens ook) doet dat met zekere regelmaat. Date's eigen visie op hoe een relationele taal er uit zou moeten zien, beschreven in 'The Third Manifesto', stelt zelfs expliciet dat implementaties van zijn ideeën niet de naam SQL mogen dragen, omdat aan die naam teveel niet-relationele associaties zouden hangen, terwijl anderzijds veel kritieken op 'het relationele model' feitelijk kritieken op SQL zijn.

Een andere manier waarop niet-relationele implementatie-aspecten kunnen doorschemeren is natuurlijk het simpelweg niet ondersteunen van bepaalde relationele mogelijkheden of features. Helemaal erg wordt het natuurlijk als bepaalde mogelijkheden, die het relationele model op zich biedt, niet worden ondersteund terwijl in plaats daarvan wel een niet-relationeel alternatief wordt geboden.

De fundamenten

Maar wat bedoelt Codd dan precies met zijn regel 0? Om dat te doorgronden moeten we terug naar datgene wat Codd met zijn relationele model wilde bereiken: een theoretisch raamwerk (dus nogmaals: geen implementatiedetails!) waarbinnen een aantal belangrijke problemen op het gebied van gegevens(bank)beheer konden worden aangepakt (Codd noemt in zijn paper uit 1969 expliciet afleidbaarheid, redundantie en consistentie).
Codd fundeerde zijn relationele theorie primair op het solide

Relational Rules (2)

De vorig jaar overleden dr. E.F. Codd werd wereldberoemd met zijn serie publicaties over een gegevens(meta)model dat later bekend zou worden onder de naam 'Relationeel Model'. De eerste uit die serie publicaties was een intern IBM Research Report dat uitkwam in 1969. 2004 zal dus gelden als een lustrum – 35 jaar Relationeel Model.
Frido van Orden schrijft in Database Magazine een serie artikelen over de betekenis en de erfenis van het gedachtengoed van Codd.

fundament van de eerste orde predikatenlogica. Het centrale concept in de relationele theorie, de relatie, is dan ook nauw verwant aan het centrale concept in de predikatenlogica, het predikaat. Voor wie zich niet laat afschrikken door droge wiskundige formules laat onderstaand voorbeeld dit zien.

Department

DEPT#	NAME	BUDGET
D1	Musical Instruments	1M
D2	Oil	10M
D3	Guns	10M

Primaire sleutel: DEPT#

Employee

EMP#	NAME	DEPT#	SALARY
1	Clinton	D1	150K
2	Bush	D2	500K
3	Johnson	D3	200K
4	Gore	D1	100K

Primaire sleutel: EMP#

Verwijzende sleutel: DEPT# → Department(DEPT#)

De betekenis van dit eenvoudige model mag voor zich spreken. Vergelijken we de relaties 'Department' en 'Employee' nu eens met de volgende predikaten:

D(x1, x2, x3): 'Er is een afdeling met ID x1, genaamd x2, met een budget x3', waarbij x1 tot en met x3 variabelen in het predikaat zijn.
E(x1, x2, x3, x4): 'Er is een employee met ID x1, genaamd x2, die werkt op afdeling x3 en een salaris heeft van x4', met x1 tot en met x4 de predikaatvariabelen.

De primaire sleutel van Department kunnen we als formule weergeven:

$\forall$ x1, x2, x3: D(x1, x2, x3) $\Rightarrow$!$\exists$ x2', x3': (x2 <> x2' | x3 <> x3') ^ D(x1, x2', x3')

Ofwel: als er een afdeling met ID x1 bestaat, naam x2 en budget x3 dan geldt dat er geen x2' en x3' te vinden zijn waarvoor geldt dat de afdeling met ID x1 deze als naam respectievelijk budget heeft, tenzij x2' en x3' identiek zijn aan x2 resp. x3. Ofwel: indien afdeling x1 een naam en budget heeft kan aan deze afdeling geen andere naam of budget worden toegekend. Er kunnen dus geen twee proposities met dezelfde ID bestaan en verschillende naam of budget.
En het geval dan waarin we de relatie Department proberen te vullen met twee identieke tupels? Welnu, hier raken we aan de essentie van het verbod op duplicaten in het relationele model. Omdat een relatie een afbeelding is van onderliggend predikaat corresponderen twee identieke tupels met twee identieke predikaten. De betekenis van het predikaat Department wordt dan bijvoorbeeld alsvolgt:

D('D1', 'Musical Instruments', '1M')
D('D2', 'Oil', '10M')
D('D3', 'Guns', '10M')
D('D3', 'Guns', '10M')

Wat is het verschil in betekenis tussen bovenstaande vier proposities en dezelfde proposities zonder de dubbele uitspraak over D3? Geen enkel natuurlijk! Chris Date verwoordt het in het Engels mooier dan ik het in het Nederlands kan bedenken: "Stating something twice does not make it more true".
Voor de verwijzende sleutel van Employee naar Department kunnen we ook een formule bedenken:

$\forall$ x1, x2, x3, x4: E(x1, x2, x3, x4) $\Rightarrow$ $\exists$ y1, y2: D(x3, y1, y2)

Ofwel: als er een employee bestaat die werkzaam is op afdeling x3 (en zeker ID, naam en salaris heeft), dan moet er een afdeling bestaan met die ID en zekere naam en budget.

Het is mogelijk om alle concepten uit het relationele model (de term 'relationele theorie' zou hier echt beter op zijn plaats zijn, maar zo heeft Codd het nu eenmaal niet genoemd) af te beelden op eerste orde predikatenlogica. De logische correctheid van het alom bekende normaliseren is bijvoorbeeld bewijsbaar met behulp van predikatenlogica. Dat zullen we hier niet doen, maar we volstaan met de (hier verder onbewezen) stelling dat alles wat in eerste orde predikatenlogica is uit te drukken, ook in het relationele model is uit te drukken.

Essentialiteit

Maar nu terug naar de oorspronkelijke vraag: waarom moet het dbms zijn werk doen met uitsluitend relationele faciliteiten?
We hebben gezien dat het relationele model een theoretisch raamwerk is voor gegevens(bank)management, gefundeerd op eerste orde predikatenlogica. Met die eerste orde predikatenlogica

> In een relationele database is de enige essentiële dataconstructie de relatie

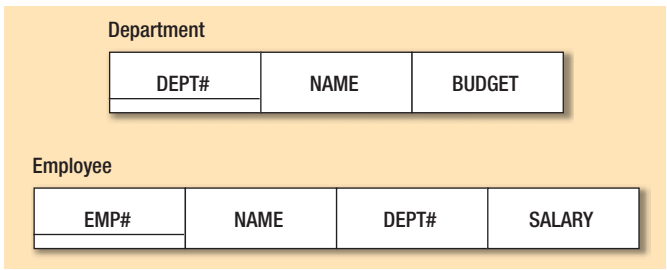
kunnen we iedere uitspraak over de werkelijkheid doen die we willen. En omdat het relationele model net zo krachtig is als eerste orde predikatenlogica, geldt hetzelfde onverkort voor het relationele model. Laat u niet misleiden door leveranciers die u anders willen doen geloven ('relationele dbms'en bieden geen ondersteuning voor rijke multimedia-gegevens', 'relationele dbms'en bieden onvoldoende faciliteiten voor datawarehouses',

etcetera): vanuit *theoretisch* oogpunt biedt een relationeel dbms alles wat u zich wensen kunt (en vanuit praktisch oogpunt meestal ook, maar dat valt buiten de invalshoek van dit artikel). En nu de crux: als u met relationele faciliteiten de wereld aankunt, waarom zou u (of het dbms zelf) dan meer nodig hebben? Iedere uitbreiding op het relationele model betekent immers dat het model niet krachtiger wordt. En waar twee oplossingen met dezelfde mogelijkheden slechts in complexiteit (aantallen concepten, operatoren en wat dies meer zij) van elkaar verschillen, dan geldt een variant op het aloude principe van Ockham's scheermes: dan verdient de eenvoudigste (dat is de minst complexe) oplossing de voorkeur.

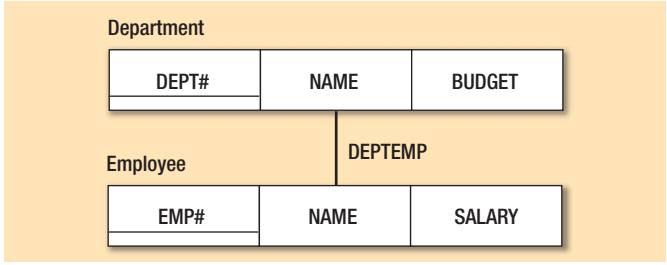
> Weglaten van een of meer concepten betekent dat het model minder krachtig wordt

Het is om deze reden dat relationele puristen als Chris Date fulmineren tegen uitbreidingen op (en nog erger: mismakingen van) het relationele model met allerhande extra faciliteiten. *Keep It Simple Stupid!*

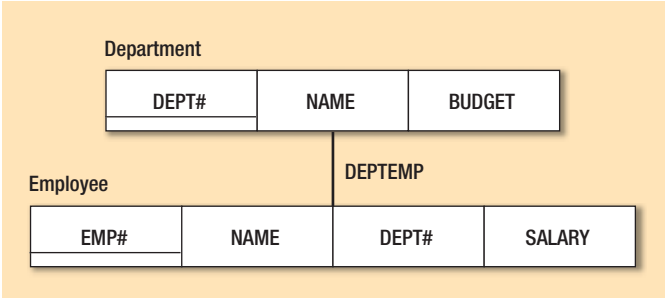
Bovenstaande redenering staat ook bekend als *het principe van essentialiteit*: binnen een model is een aantal concepten essentieel. Weglaten van een of meer concepten betekent dat het model minder krachtig wordt. Omgekeerd betekent het toevoegen van extra concepten echter niet dat het model krachtiger wordt, er ontstaan slechts meer mogelijkheden om hetzelfde uit te drukken. Daarmee wordt het model in de praktijk zelfs vaak minder krachtig: door de verschillende manieren om hetzelfde uit te drukken wordt interpretatie moeilijker in plaats van makkelijker. Het principe van essentialiteit laat zich goed illustreren met behulp van het reeds gehanteerde Department / Employee voorbeeld. We zullen dit voorbeeld modelleren in zowel het relationele als het netwerkmodel.



Afbeelding 1: Relationeel model.



Afbeelding 2: Netwerkmodel.



Afbeelding 3: Het nieuwe model.

In het relationele model zien we de welbekende relaties Department en Employee. In het netwerkmodel vinden we het concept van een 'CODASYL set'. Iedere *occurrence* in die CODASYL set bestaat uit een Department record, een verzameling bijbehorende Employee records en een *link* ("DEPTEMP") die de Department en Employee records aan elkaar koppelt. Binnen een CODASYL set occurrence kan de link worden beschouwd als een pointer-ketting: een pointer van het Department record naar het eerste Employee record voor dat Department, een pointer van dat Employee record naar het volgende Employee record voor dat Department, enzovoort, en tot slot een pointer van het laatste Employee record terug naar het Department.
In het netwerkmodel is te zien dat de Employee records geen DEPT# attribuut bevatten. Indien we uit willen vinden tot welk Department zekere Employee behoort, moeten we de DEPTEMP link occurrence doorlopen van het Employee record tot het corresponderende Department record. Omgekeerd moeten we om de Employees die op zeker Department werken te vinden, de DEPTEMP link occurrence doorlopen van het Department record over alle Employee records. Met andere woorden: de informatie die in het relationele model door middel van een verwijzende sleutel wordt gerepresenteerd, wordt in het netwerkmodel als een link gerepresenteerd.

We beschouwen nu twee query's:
Query 1.

```
SELECT EMP#, NAME
FROM Employee
WHERE SALARY > 100K
```

Query 2.

```
SELECT EMP#, NAME
FROM Employee
WHERE SALARY > 100K
  AND DEPT# = 'D3'
```

In het relationele model zijn deze query's wat betreft de structuur identiek. Indien we de query's echter afbeelden op het netwerk-model zullen we voor Query 2 op een of andere wijze moeten uitdrukken dat we de DEPTEMP link moeten doorlopen. Als deze en soortgelijke vragen vaker voorkomen ligt het natuurlijk voor de hand om het attribuut DEPT# gewoon op te nemen in het Employee record, waarmee het model in afbeelding 3 ontstaat.
Nu vallen twee zaken op: de relationele en netwerk-query's worden identiek en hetzelfde geldt voor de beide modellen, met uitzondering van de DEPTEMP link. Het is echter ook duidelijk dat het nieuwe model redundantie bevat: de DEPTEMP link en het attribuut DEPT# in Employee drukken hetzelfde uit, namelijk dat zekere Employee op zekere Department werkzaam is. Een van beide kan dus vervallen, maar welke is essentieel? Voor de hand ligt om te zeggen 'het DEPT# attribuut', dat hebben we immers zelf net toegevoegd. Toch is dit niet correct.
Als we naar het relationele model in afbeelding 1 kijken zien we dat alle dataconstructies (alle rijen en alle kolommen) essentieel zijn. Hetzelfde geldt voor het originele netwerkmodel uit afbeelding 2: alle rijen, kolommen en links zijn essentieel. Maar in het aangepaste netwerkmodel uit afbeelding 3 is de link niet meer essentieel: er is geen enkele informatie die uit het netwerk (inclusief link) kan worden gehaald die niet afleidbaar is uit rijen en kolommen alleen (zie het originele relationele model dat gelijk is aan het aangepaste netwerkmodel, minus de link). Hiermee hebben we ook een essentieel verschil ontdekt tussen relationele en andere databases (hiërarchische, netwerk-, object-georiënteerde of wat dan ook). In een relationele database is de enige essentiële dataconstructie de relatie. In andere databases is er tenminste één andere essentiële dataconstructie, meestal in de vorm van een of andere vorm van verwijzing (link, reference, pointer). We kunnen het ook anders stellen: iedere database die geen andere essentiële dataconstructie bevat dan een relatie (of tabel) is feitelijk te beschouwen als een relationele database. Immers, iedere niet-essentiële dataconstructie is geen onderdeel van het logisch model van de database en kan derhalve worden beschouwd als een implementatie-detail.

Conclusie

Met Codd's regel 0 hebben we een tweetal grondprincipes van het relationele model verkend die we in latere bijdragen in deze serie nog vaker tegen zullen komen:

- De scheiding tussen model (= theorie) en implementatie;
- De essentialiteit van concepten en constructies.

De volgende keer zullen we met regel 1 hierop doorgaan.

Frido van Orden (frido.van.orden@faapartners.com) is partner bij FAA Partners.