De regels van Codd allerminst verworden tot dooddoeners

Relationele producten niet meer weg te denken

Frido van Orden

Dit is de laatste aflevering uit de serie Relational Rules. Deze keer worden de laatste drie van Codd's 12 regels behandeld. Last maar zeker niet least, want we zullen de gevestigde relationele producten nog wel even stevig op de pijnbank leggen.

Een van de belangrijkste zaken die een DBMS voor u regelt, of dat nu een relationeel DBMS is of niet, is het waarborgen van de integriteit van de gegevens die in de database zijn opgeslagen. Het is dan ook niet verwonderlijk dat Codd een van zijn 12 regels weidt aan het vastleggen van integriteitregels.

Relational Rules (10 en slot)

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. Frido van Orden besluit zijn serie artikelen over de betekenis en de erfenis van het gedachtengoed van Codd. In deze laatste aflevering worden de regels 10, 11 en 12 besproken.

Regel 10: Integriteit onafhankelijkheid.
Integriteitregels dienen te worden gespecificeerd separaat van applicatieprogramma's en te worden opgeslagen in de catalogus. Het moet mogelijk zijn dergelijke regels te veranderen wanneer dat wenselijk is, zonder bestaande applicaties onnodig te beïnvloeden.

Regel 11: Distributie onafhankelijkheid.
Bestaande applicaties dienen te blijven werken:
a. indien een gedistribueerde versie van het DBMS wordt geïntroduceerd;
b. indien bestaande gedistribueerde gegevens worden herdistribueerd over het systeem.

Regel 12: De non-ondermijning regel.
Indien het systeem een laag niveau ('per-record') interface ondersteunt, dan kan de interface niet worden gebruikt om het systeem te ondermijnen, bijvoorbeeld door een relationele beveiliging- of integriteitregel te passeren.

Integriteit

Laten we om te beginnen definiëren wat we precies bedoelen met integriteitregels. Integriteitregels komen in vele smaken:
- verplichtingsregels (bijvoorbeeld Med# is een verplicht attribuut van Medewerker);
- domeinregels (bijvoorbeeld Med# is een numeriek veld van zes posities);
- uniciteitsregels (Med# is uniek over alle tupels in Medewerker);
- verwijzende sleutel regels (waarden in Afd# in Medewerker moeten verwijzen naar een bestaande Afd# waarde in Afdeling);
- attribuutregels (bijvoorbeeld Geslacht in Medewerker heeft als geldige waarden M en V);
- tupelregels (bijvoorbeeld 'Datum in dienst' in Medewerker moet voor 'Datum uit dienst' liggen);
- databaseregels (bijvoorbeeld Budget in Afdeling moet groter zijn dan de som van de Salarissen van alle Medewerkers die op de Afdeling werkzaam zijn).

> Op een aantal wezenlijke punten vallen vrijwel alle relationele producten van vandaag door de mand

Volgens Codd moeten alle integriteitregels in de database worden gespecificeerd en niet in de applicatie. Het kan dan natuurlijk niet zo zijn dat er bepaalde integriteitregels niet door de database worden ondersteund. Met andere woorden: regel 10 legt aan ieder relationeel DBMS de eis op dat alle vormen van integriteitregels worden ondersteund!
In de SQL standaard is het allemaal netjes geregeld: vanaf SQL92 worden alle vormen van integriteitregels ondersteund. Veelal gebeurt dit door middel van specifieke taalconstructies als 'NOT NULL', 'ADD CONSTRAINT PRIMARY KEY' of 'ADD CONSTRAINT FOREIGN KEY ... REFERENCES'.
Daarnaast bestaat er echter het algemene concept van *assertions*, waarmee willekeurige regels op de database kunnen worden gedefinieerd, zoals bijvoorbeeld:

```
CREATE ASSERTION min_budget CHECK ( NOT EXISTS
(SELECT 42
FROM Afdeling
WHERE Budget > (SELECT SUM(Salaris)
        FROM Medewerker
        WHERE Medewerker.Afd# = Afdeling.Afd#
        )
)
)
```

Natuurlijk kunnen met assertions ook verplichtingsregels, verwijzende sleutel regels etcetera. worden gespecificeerd. Wat betreft betekenis maakt het niets uit, de specifieke taalconstructies zijn echter meestal eenvoudiger, beter leesbaar en minder foutgevoelig om te specificeren.

Hoewel assertions voor informatica-begrippen al een eeuwigheid in de SQL standaard worden ondersteund, schitteren ze echter tot de dag van vandaag in alle relationele DBMS'en (geen enkele uitgezonderd!) door afwezigheid. De leveranciers, die gezamenlijk de SQL standaardisatie bepalen, zijn dan ook zo slim geweest om assertions niet in het Entry level van SQL92 te stoppen. Vervolgens kan iedereen claimen dat Entry level SQL-92 door het eigen product wordt ondersteund naast zeer vele elementen uit andere levels van de specificatie.

Wat wel door vrijwel ieder relationeel DBMS wordt ondersteund zijn 'check constraints'. Net als assertions maken check constraints het mogelijk om een SQL-expressie als integriteitregel te definiëren. De beperkingen die aan de SQL-expressie worden gesteld komen er echter in de praktijk op neer dat check constraints alleen kunnen worden gebruikt om attribuut- of tupelregels te specificeren.
Een tweetal voorbeelden:

```
ALTER TABLE Medewerker
ADD CONSTRAINT chk_gsl
CHECK(geslacht IN ('M','V'));

ALTER TABLE Medewerker
ADD CONSTRAINT chk_dat
CHECK(dat_in_dienst < dat_uit_dienst OR
                          dat_uit_dienst IS NULL);
```

Het is beter dan niets, maar het blijft natuurlijk uitermate treurig dat zelfs het meest luxueuze relationele DBMS dat u vandaag kunt kopen niet in staat is om integriteitregels af te dwingen die betrekking hebben op meer dan 1 record, unieke en verwijzende sleutels daargelaten. En daarmee is regel 10 alleen al voldoende om, volgens de 30 jaar oude regels van Codd, alle zichzelf relationeel noemende database-producten af te serveren! Daar moet u het de volgende keer toch eens met uw Oracle, IBM of Sybase account manager over hebben.

Distributie

Regel 11 is wat mij betreft de minst interessante van al Codd's regels. De regel stelt dat het distribueren van het DBMS en de gegevens in de database over meerdere systemen geen invloed mag hebben op applicaties. Deze zaken zijn echter niets anders dan een speciaal geval van het concept van fysieke gegevens-onafhankelijkheid, waarover regel 8 reeds algemene uitspraken doet.

Geen achterdeurtjes

Regel 12 is zonder meer een van de leukste uitsmijters die Codd voor zijn lijstje had kunnen verzinnen. Feitelijk is het niet veel meer dan een herhaling van regel 0, de regel die twee grondprincipes van het relationele model introduceerde:
- de scheiding tussen model (= theorie) en implementatie;
- de essentialiteit van concepten en constructies.
Voor wie na het bestuderen van regel 0 tot en met 11 nog de illusie had dat een relationele database te implementeren zou zijn als een 'syntactisch suikerlaagje' bovenop andere technologie, wordt met regel 12, op zich ten overvloede, alsnog de deur dichtgeslagen. Het relationele model definieert op een modelmatige en implementatie-onafhankelijke wijze hoe een database er uitziet. Op het moment dat de database ook op niet-relationele wijze te benaderen zou zijn (formeel spreekt regel 12 over een 'per-record' interface, maar in het algemeen wordt een interface bedoeld die toegang biedt tot de gegevens in de database buiten de relationele interface om) kan het relationele DBMS niet meer garanderen dat de gegevens in de database voldoen aan het database-model dat is gedefinieerd. In het beste geval zou het RDBMS de fouten in de database kunnen detecteren, die dan uiteraard buiten de relationele interface om weer gerepareerd zouden moeten worden.

> Bij het opnieuw leggen van de verwijzende sleutel wordt deze gecontroleerd

Uiteraard weten productleveranciers hier weer creatief mee om te gaan. Een leuk voorbeeld is het 'TRUNCATE TABLE' feature dat door een aantal DBMS'en wordt ondersteund. Het doel van TRUNCATE TABLE is kortgezegd om snel alle records uit een tabel te kunnen verwijderen. Hierbij worden bijvoorbeeld controles omzeild op verwijzende sleutels, die mogelijk geschonden zouden kunnen worden. Volgens regel 12 is dergelijke functionaliteit verboden, omdat in dit geval een integriteitregel geschonden kan worden. De oplossing die veelal geboden wordt is dat het uitvoeren van TRUNCATE TABLE tot een foutmelding leidt indien er verwijzende sleutels naar de tabel in kwestie zijn gedefinieerd. Pas als deze verwijzende sleutels zijn verwijderd, of uitgeschakeld, kan de tabel door middel van TRUNCATE TABLE

worden geleegd. Bij het opnieuw leggen van de verwijzende sleutel, of het weer inschakelen ervan, wordt de verwijzende sleutel gecontroleerd en volgt er een foutmelding indien de verwijzende sleutel geschonden is.

Tot slot

Vijfentwintig jaar geleden formuleerde Codd een dertiental regels waaraan relationele producten zouden moeten voldoen om het label 'relationeel' te mogen dragen. Inmiddels zijn relationele producten niet meer weg te denken en is het eind van de relationele technologie nog bij lange na niet in zicht.
Hoe vanzelfsprekend het relationele model nu ook lijken mag, we hebben in deze serie artikelen gezien dat de regels van Codd allerminst zijn verworden tot dooddoeners. Op een aantal wezenlijke punten vallen alle of vrijwel alle relationele producten van vandaag door de mand.
Blijkbaar zijn zelfs deze kreupele, halfbakken, pseudo-relationele producten zo goed dat ze de concurrentie van oude en nieuwe niet-relationele technologie met gemak aankunnen. Een intrigerende gedachte voor de toekomst.

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