Tel- en kopieerregels in het Rules Capable RDBMS

De Business Rules Approach (3)

Marc Dierick

In een eerste artikel in DB/M 6, 2008 werd een beeld geschetst betreffende business rules en de business rules approach. Er werd gesteld dat de business rules approach niet haaks staat op het relationele model, maar dat het relationele gegevensmodel samen met het procesmodel en het bedrijfsregelmodel een belangrijk onderdeel vormt bij het toepassen van de business rules approach.

Daarbij wordt het relationele gegevensmodel verrijkt met een aantal bedrijfsregelspecifieke gegevenselementen, wat leidt tot een rule-enriched gegevensmodel[1]. In een tweede artikel in DB/M 4, 2009 werd een formele beschrijving gegeven van dit model in een uitgebreide relationele catalogus. Doel van dit derde artikel is het rule-enriched gegevensmodel verder uit te breiden met een aantal constructs die handig zijn bij het afdwingen van bedrijfsregels, namelijk tel- en kopieerregels.
We hernemen het voorbeeld uit de vorige twee artikelen. Het gaat hierbij om een variant van een al eerder in DB/M gepubliceerd voorbeeld[2] betreffende een organisatiestructuur, waarbij de organisatie bestaat uit afdelingen en diensten. Medewerkers kunnen tewerkgesteld worden op niveau van de organisatie, binnen afdelingen en binnen diensten, waarbij het mogelijk is tegelijkertijd op verschillende niveaus tewerkgesteld te worden. Een voorbeeld van een tewerkstelling op het niveau van de organisatie is die van de algemeen directeur, maar ook zijn secretaresse fungeert op dat niveau. Op niveau van afdeling hebben we de afdelingschef, maar ook daar diens secretaresse. Hetzelfde geldt voor een werknemer die toegewezen wordt aan een dienst. Die verricht prestaties op niveau van deze dienst. Uiteindelijk betreft het hier het merendeel van de werknemers die werkzaam zijn binnen de organisatie. In de vorige artikelen zijn sommeringregels en formules beschreven als kennisregels die opgenomen kunnen worden in een RCRDBMS. Indien men alleen over deze types van kennisregels beschikt komt men echter niet ver.

Introductie telregels

Stel dat we een bijkomende bedrijfsregel hebben waarbij aangegeven wordt dat minstens 40 procent van de medewerkers die tewerkgesteld zijn binnen de organisatie vrouw dient te zijn. Dan gaan we niet meer sommeren, maar tellen. Om deze bedrijfsregel af te dwingen voegen we eerst een kolom toe aan tabel MEDEWERKER:

```
ALTER TABLE MEDEWERKER
(
        ADD     GESLACHT CHAR(1) NOT NULL,
);
```

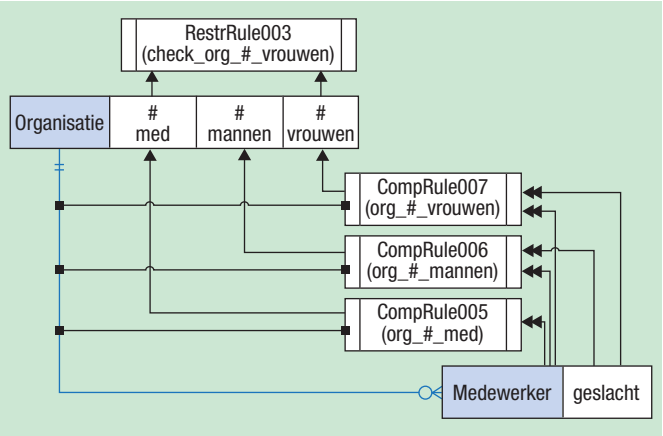
De waarde van kolom geslacht moet ingevuld worden en mag slechts de waarden 'M' of 'V' aannemen. Dit regelen we in met behulp van een not null en een column constraint.
De bedrijfsregel betreffende de verhouding tussen mannen en vrouwen is van toepassing op de gehele organisatie. We voegen daarom twee virtuele kolommen AANT_MANNEN en AANT_VROUWEN toe aan tabel ORGANISATIE.

```
ALTER TABLE ORGANISATIE
(
        ADD     AANT_MANNEN INT VIRTUAL,
                AANT_VROUWEN INT VIRTUAL
);
```

Vervolgens berekenen we de waarde van deze kolommen op basis van de informatie die we terugvinden in tabel MEDEWERKER. Juist zoals het geval is bij sommeren, gebeurt tellen ook steeds over een verwijzende sleutel heen. Tussen de tabellen ORGANISATIE en MEDEWERKER ligt echter geen verwijzende sleutel. Zonder een kolom ORG_ID toe te voegen aan tabel MEDEWERKER is het in een RDBMS ook niet mogelijk om die verwijzende sleutel te definiëren.
Het toevoegen van die kolom zal echter op heel wat tegenstand botsen bij DBA's. Immers, tabel ORGANISATIE bevat slechts één tupel. We zouden in elk MEDEWERKER-tupel naar dezelfde organisatie verwijzen, terwijl we impliciet weten om welke organisatie – er is er maar één – het gaat.

Een nieuw type verwijzende sleutel

Eén en ander heeft tot gevolg dat we in een RCRDBMS iets dienen te wijzigen aan de definitie van een verwijzende sleutel. In een RDBMS geven we aan dat één of meer kolommen uit een child-tabel verwijzen naar één of meer kolommen uit een parent-tabel. Kolommen uit de child-tabel en de parent-tabel worden paarsgewijze aan elkaar gekoppeld door middel van een verwijzende sleutel.



Afbeelding 1: Bedrijfsregel verhouding tussen mannen en vrouwen.

Indien een parent-tabel slechts één tupel bevat is het echter overbodig child-kolommen naar parent-kolommen te laten verwijzen. Eerst wijzigen we tabel ORGANISATIE, teneinde aan te geven dat die juist één tupel moet bevatten:

```
ALTER TABLE ORGANISATIE SET AS SINGLETON;
```

Dan definiëren we een verwijzende sleutel tussen MEDEWERKER en ORGANISATIE:

```
ALTER TABLE MEDEWERKER
        ADD CONSTRAINT FK_ORG_MED
        FOREIGN KEY
        REFERENCES ORGANISATIE;
```

Dit DDL-statement is alleen succesvol als de parent-tabel maximaal één tupel kan bevatten. Voor de rest werkt de verwijzende sleutel op dezelfde wijze als verwijzende sleutels die verwijzen naar parent-tabellen die meerdere tupels kunnen bevatten. Bij het toevoegen van een medewerker wordt gecontroleerd of de organisatie bestaat.
En bij het verwijderen van de organisatie wordt gecontroleerd of er geen medewerkers zijn. Het wijzigen van waarden van child-kolommen is niet mogelijk omdat die er niet zijn, en het wijzigen van de primaire sleutel van de organisatie heeft ook al geen gevolgen.

Deze wijzigingen behoeven dus geen controle, terwijl dit bij gangbare verwijzende sleutels wel het geval is.

Terug naar de telregels

Nu deze ietwat speciale verwijzende sleutel geformaliseerd is kunnen we overgaan tot de telregels die de berekening van de kolommen ORG.AANT_MANNEN en ORG.AANT_VROUWEN bewerkstelligen.
Als voorbeeld de definitie van de telregel voor kolom ORG.AANT_MANNEN:

```
ALTER FOREIGN KEY FK_ORG_MED
(
        ADD CALCULATION RULE ORG_AANT_MANNEN
        COMPUTE ORG.AANT_MANNEN
        AS NUMBER OF ALL RELATED MEDEWERKER RECORDS
        WHERE MED.GESLACHT = "M"
);
```

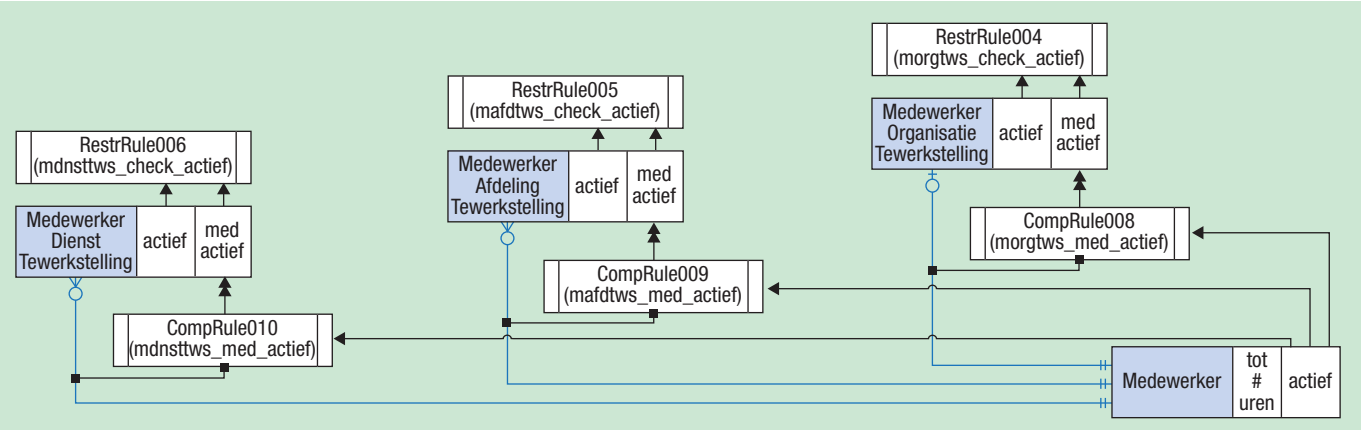
Voor kolom ORG_AANT_VROUWEN voegen we een gelijkaardige telregel toe. De beperkingregel die moet controleren of het aantal vrouwen dat tewerkgesteld is binnen de organisatie minstens 40 procent bedraagt van het totaal aantal medewerkers, wordt dan gedefinieerd als gewone check constraint op tabel ORGANISATIE:

```
ALTER TABLE ORGANISATIE
        ADD CONSTRAINT CHECK_AANT_VROUWEN
        CHECK   AANT_VROUWEN *100 / (AANT_MANNEN +
                                        AANT_VROUWEN)
        >= 40;
```

Zoals in het eerste artikel aangegeven dienen beperkingregels echter genormaliseerd te worden. Bovenstaande check constraint is allesbehalve atomair.

We normaliseren de beperkingregel

Om de beperkingregel te normaliseren nemen we een kolom AANT_MED op in tabel ORGANISATIE, waarvan de waarde gelijk is aan het totaal aantal medewerkers. De waarde van deze kolom kan op twee manieren berekend worden. Als som



Afbeelding 2: Uitgebreid gegevensdiagram.

van de waarden van de kolommen AANT_MANNEN en AANT_VROUWEN, of als telregel over de verwijzende sleutel tussen de tabellen ORGANISATIE en MEDEWERKER heen. Mijns inziens leidt deze laatste oplossing tot minder onderhoud bij wijzigende systeemvereisten:

```
ALTER TABLE ORGANISATIE
(
        ADD     AANT_MED INT VIRTUAL
);
ALTER FOREIGN KEY FK_ORG_MED
(
        ADD CALCULATION RULE ORG_AANT_MED
        COMPUTE ORG.AANT_MED
                AS NUMBER OF ALL RELATED MEDEWERKER
                                                RECORDS);
```

In tegenstelling tot de vorige telregels is deze telregel niet conditioneel. We kunnen de check constraint nu herschrijven als:

```
ALTER TABLE ORGANISATIE
        DROP CONSTRAINT CHECK_AANT_VROUWEN;
ALTER TABLE ORGANISATIE
        ADD CONSTRAINT CHECK_AANT_VROUWEN
        CHECK   AANT_VROUWEN *100 / AANT_MED
                >= 40;
```

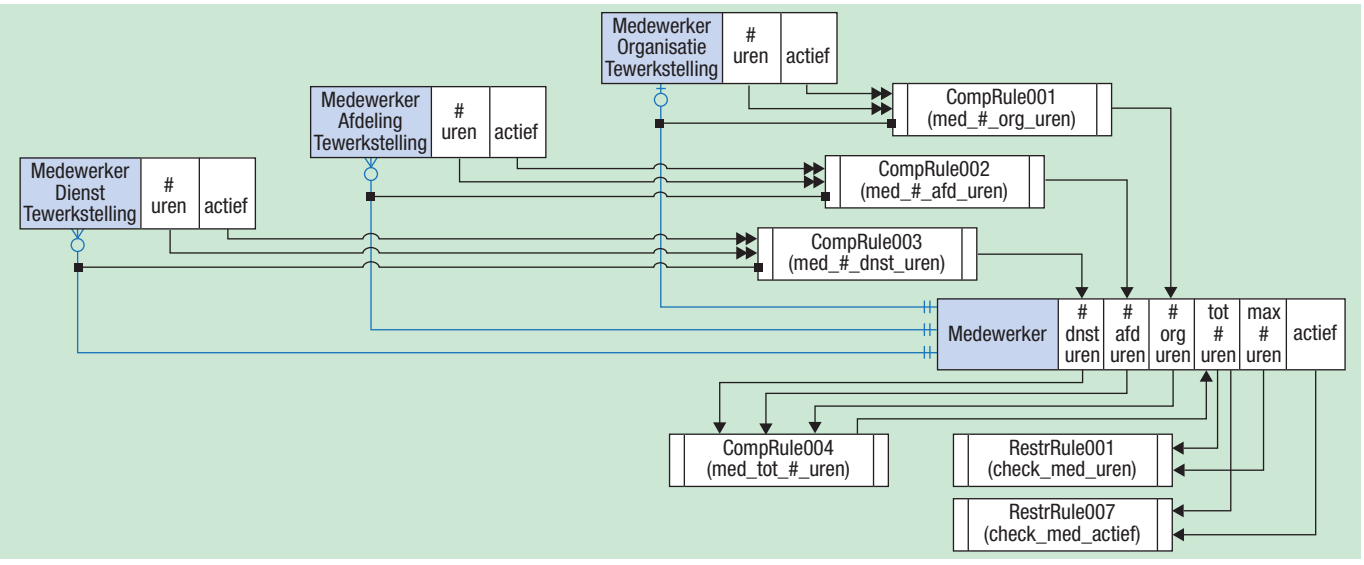
Het gegevensmodel wordt uitgebreid met bedrijfsregels die opgenomen worden in de uitgebreide catalogus van de gegevensbank. De bedrijfsregel betreffende de verhouding tussen mannen en vrouwen is te zien in afbeelding 1.
Het RCRDBMS is in staat zijn gepersisteerde gegevens consistent te houden met deze DDL-definities. Wat betreft telregels dient het RCRDBMS in staat te zijn zelf te bepalen wanneer deze uitgevoerd dienen te worden. Voor de telregels van zowel het totaal aantal medewerkers, het aantal tewerkgestelde mannen en het aantal tewerkgestelde vrouwen dient het RCRDBMS deze waarden te (her)berekenen in de volgende gevallen:
1. Bij het lezen van een ORGANISATIE-tupel;
2. Bij het toevoegen van een MEDEWERKER-tupel;
3. Bij het verwijderen van MEDEWERKER-tupel.

Voor de berekening van het aantal mannen en het aantal vrouwen dient het RCRDBMS bovendien rekening te houden met het feit dat deze telregels conditioneel zijn. Ze hangen af van het geslacht van de medewerker. Deze berekeningen dienen dus ook uitgevoerd te worden in het volgende geval:
4. Bij het wijzigen van MEDEWERKER.GESLACHT.

Telregels kunnen uiteraard ook gedefinieerd worden over verwijzende sleutels heen waarvan de parent-tabel meer dan één tupel kan bevatten. Het wordt aan de lezer overgelaten te bepalen wanneer het RCRDBMS de berekening van zo'n telregel dient uit te voeren. Overigens kan men ook sommeringregels definiëren die conditioneel zijn. Zo zou men het totaalbudget aan salarissen van mannen en het totaalbudget aan salarissen van vrouwen kunnen berekenen op niveau van de organisatie.

Introductie kopieerregels

Buiten sommeringregels, telregels en formules is er nog een belangrijk type van kennisregels, namelijk kopieerregels.
Stel dat we medewerkers en tewerkstellingen op inactief kunnen zetten. Dan willen we dat het voor een inactieve medewerker onmogelijk wordt deze nog toe te wijzen aan de organisatie, afdelingen of diensten om aldaar tewerkgesteld te worden. Maar we willen ook niet dat een medewerker op inactief gezet kan worden indien er nog tewerkstellingen zijn die actief zijn.
We voegen een gepersisteerde kolom ACTIEF toe aan de tabellen MEDEWERKER, MED_ORG_TEWERKSTELLING, MED_AFD_TEWERKSTELLING en MED_DNST_TEWERKSTELLING. Aan de tewerkstellingstabellen voegen we bovendien een virtuele kolom MED_ACTIEF toe. Doel is deze kolom te allen tijde dezelfde waarde als die van de gerelateerde kolom MEDEWERKER.ACTIEF te laten aannemen.



Afbeelding 3: Gegevensmodel.

Voor organisatietewerkstellingen kunnen we dan een check constraint schrijven waarbij MORGTWS.ACTIEF alleen de waarde "J" kan aannemen als ook MORGTWS.MED_ACTIEF de waarde "J" heeft.

```
ALTER TABLE MED_ORG_TEWERKSTELLING
        ADD CONSTRAINT CHECK_ACTIEF
        CHECK NOT (ACTIEF = "Y" AND MED_ACTIEF = "N");
```

Voor afdeling- en diensttewerkstellingen hebben we gelijkaardige check constraints. Wel moeten we de berekening van MED_ACTIEF-kolommen nog definiëren in het RCRDBMS.
Als voorbeeld nemen we kolom MAFDTWS.MED_ACTIEF:

```
ALTER FOREIGN KEY FK_MED_MAFDTWS
(
        ADD CALCULATION RULE MAFDTWS_MED_ACTIEF
        COMPUTE MAFDTWS.MED_ACTIEF AS COPY OF
                                    RELATED MED.ACTIEF
);
```

Het herberekenen van de waarde van kolom MAFDTWS.MED_ACTIEF dient te gebeuren in de volgende gevallen:

1. Bij het lezen van een MED_AFD_TEWERKSTELLING-tupel;
2. Bij het toevoegen van een MED_AFD_TEWERKSTELLING tupel;
3. Bij het wijzigen van MED_AFD_TEWERKSTELLING.MED_ID;
4. Bij het wijzigen van MEDEWERKER.ACTIEF.

Deze regels kunnen voorgesteld worden door middel van een uitgebreid gegevensdiagram, zoals in afbeelding 2.

Nu we weten wat sommeringregels, telregels, kopieerregels en formules zijn is de basis gelegd voor een geformaliseerd rule-enriched gegevensmodel. Check constraints, verwijzende sleutels en zo meer kennen we al als onderdelen van het relationele gegevensmodel. Rest ons – zoals ook het geval is bij toepassing van design patterns in objectoriëntatie – de verschillende types kennisregels te combineren om zo te komen tot een opsplitsing van complexe beperkingregels in deze eerder eenvoudige kennis- en beperkingregels.
Als voorbeeld van het combineren van kennisregels maken we de berekening van de kolommen MED.AANT_ORG_UREN, MED.AANT_AFD_UREN en MED.AANT_DNST_UREN afhankelijk van de activiteitsstatus van de tewerkstellingtupels. Dit houdt in dat we de sommeringregels uit het vorige artikel conditioneel maken. Zo voegen we aan de sommeringregel die MED.AANT_DNST_UREN berekent de conditie WHERE MDNSTTWS.ACTIEF = "J" toe. Als kers op de taart kunnen we dan nog een check constraint toevoegen die het onmogelijk maakt een medewerker op inactief te zetten indien het totaal aantal uren dat die medewerker tewerkgesteld is groter is dan 0. Zie afbeelding 3.
Het aanpassen van de 60/40 procentregel waarbij aangegeven wordt dat minstens 40 procent van de actieve medewerkers die tewerkgesteld zijn binnen de organisatie vrouw dient te zijn, blijkt ook een koud kunstje (in de oorspronkelijke regel wordt geen rekening gehouden met het al dan niet actief zijn van medewerkers).
Een RCRDBMS laat toe een behoorlijk complex systeem volledig declaratief te definiëren, tenminste wat betreft de bedrijfsregels. Zoals blijkt uit de tekst hoeft men ook niet alle regels te kennen alvorens men aan de bouw van een administratief informatiesysteem begint. Het RCRDBMS laat toe een systeem op iteratieve wijze op te bouwen. De rol van de DBA wordt daarbij terug opgewaardeerd. Hij (of is het een zij? Zie de 40 procent-regel) krijgt meer controle over de kwaliteit van de informatie die opgeslagen wordt in de gegevensbank. Misschien wordt hij wel opgewaardeerd tot een RDBA (Rules & Data Base Administrator)?

Conclusie

In dit artikel wordt een fictief systeem beschreven dat gezien kan worden als een RCRDBMS (rules capable RDBMS), waaraan de functionaliteit van telregels en kopieerregels is toegevoegd.
Het toevoegen van tel- en kopieerregels laat ons toe nog meer kennisregels te vangen dan al het geval was met enkel sommeringregels en formules.
Zoals aangegeven in de vorige artikelen worden bedrijfsregels in de vorm van complexe volzinnen omgezet in kleinere samenhangende kennisregels (formules, sommering-, tel-, kopieerregels) en beperkingregels. Elk van deze regels wordt op declaratieve wijze gedefinieerd binnen het RCRDBMS. Op welke informatie de regels inspelen en wanneer, wordt automatisch bepaald door het RCRDBMS. De DBA krijgt zo meer controle over de kwaliteit van de informatie die zich in de gegevensbank bevindt.
Men komt tot hergebruik van regels en meer nog tot hergebruik van generieke applicatiecode. Wat we ook willen sommeren, steeds wordt dezelfde generieke applicatiecode gebruikt om tot de berekening van een som te komen. Hetzelfde geldt voor de tel- en kopieerregels. Met verwijzende sleutels in een gangbaar RDBMS gaat het overigens net zo. Bij het afdwingen van referentiële integriteit wordt steeds gebruik gemaakt van dezelfde generieke applicatiecode binnen de database engine.
Hergebruik ten top!
Als we de aanhangers mogen geloven, zou deze aanpak moeten zorgen voor informatiesystemen die robuuster, flexibeler en onderhoudsarmer zijn en die beter voldoen aan de noden van de eindgebruikers. De keuze is gemakkelijk. Ofwel een RCRDBMS aankopen, ofwel er één bouwen. Of geen van beide, wat overeenstemt met een keuze waarbij bedrijfsregels verborgen blijven in applicatiecode, (onvolledige) documentatie en in de hoofden van de eindgebruikers zelf.

Noten

1. *Barbara von Halle, Business Rules Applied, Building Better Systems Using the Business Rules Approach.*
2. *R.J. Veldwijk, 10 Geboden voor Goed Database-Ontwerp.*

Marc Dierick (marc@mardinfo.be) is onafhankelijk IT-consultant bij Mardinfo.