Een introductie tot het Rules Capable RDBMS

De Business Rules Approach (2)

Marc Dierick

In een eerder artikel in Database Magazine 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]. Doel van dit tweede artikel is aan te geven hoe men een rule-enriched gegevensmodel formeel kan beschrijven in een uitgebreide relationele catalogus. We hernemen het voorbeeld uit het vorige artikel, gepubliceerd in DB/M 6, 2008.

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 algemene directeur, maar ook diens 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 die dienst. Uiteindelijk betreft het hier het merendeel van de werknemers die werkzaam zijn binnen de organisatie.

We maken nu de stap naar een formele definitie van het rule-enriched gegevensmodel dat beschreven werd in het vorige artikel in DB/M 6, 2008. Hierbij wordt gekozen voor een fictieve DDL-taal als uitbreiding op de relationele DDL-taal, zoals voorzien in de meeste RDBMS'en. Wat betreft het creëren van tabellen, indexen en sleutels wijzigt er omzeggens niets ten aanzien van het definiëren van deze objecten in een standaard RDBMS. Wel dient er een uitbreiding voorzien te worden voor virtuele (lees niet gepersisteerde) tabellen en kolommen die deel uitmaken van het rule-enriched gegevensmodel. We dienen immers de mogelijkheid te hebben tabellen en kolommen te creëren die 'berekend' worden door middel van kennisregels op basis van gepersisteerde informatie.

We dienen de volgende tabellen te creëren:

ORGANISATIE, de tabel waarin de informatie betreffende de organisatie wordt opgenomen;

AFDELING, de tabel waarin de informatie betreffende afdelingen wordt opgenomen;

DIENST, de tabel waarin informatie betreffende diensten wordt opgenomen;

MEDEWERKER, de tabel waarin informatie betreffende medewerkers wordt opgenomen;

MED_ORG_TEWERKSTELLING, de tabel waarin informatie betreffende tewerkstellingen op niveau van de organisatie wordt opgenomen;

MED_AFD_TEWERKSTELLING, de tabel waarin informatie betreffende tewerkstellingen op niveau van afdelingen wordt opgenomen;

MED_DNST_TEWERKSTELLING, de tabel waarin informatie betreffende tewerkstellingen op niveau van diensten wordt opgenomen.

Als voorbeeld geven we hierbij de DDL-statements ter creatie van de tabellen MED_AFD_TEWERKSTELLING en MEDEWERKER. Per tabel wordt ook een alias meegegeven. Dat is nadien handig bij het definiëren van de bedrijfsregels. Twee voorbeelden, waaronder een voorbeeld met virtuele kolommen:

```
CREATE TABLE MED_AFD_TEWERKSTELLING MAFDTWS
(
        MED_ID          INT,
        AFD_ID          INT,
        AANT_UREN       NUMBER
);

CREATE TABLE MEDEWERKER MED
(
        ID              INT,
        NAAM            VARCHAR(50),
        VOORNAAM        VARCHAR(50),
        SALARIS         NUMBER,
        MAX_AANT_UREN   NUMBER,
        AANT_ORG_UREN   NUMBER VIRTUAL,
        AANT_AFD_UREN   NUMBER VIRTUAL
```

```
        AANT_DNST_UREN NUMBER VIRTUAL,
        TOT_AANT_UREN  NUMBER VIRTUAL
)
```

Het creëren van primaire sleutels, verwijzende sleutels en indexen gebeurt op dezelfde wijze als in een 'normaal' RDBMS. Waar het echt leuk wordt is bij de definitie van de kennisregels ten behoeve van de virtuele kolommen die toegevoegd zijn aan het gegevensmodel. Bij het definiëren van deze kennisregels gaan we ervan uit dat de verwijzende sleutels FK_MED_MORGTWS, FK_MED_MAFDTWS en FK_MED_MDNSTWS tussen de tabel MEDEWERKER en respectievelijk de tabellen MED_ORG_TEWERKSTELLING, MED_AFD_TEWERKSTELLING en MED_DNST_TEWERKSTELLING eerst gedefinieerd worden.

In ons voorbeeld hebben we twee categorieën van kennisregels, namelijk sommeringregels en formules. Aan de hand van sommeringregels worden de kolommen MED.AANT_ORG_UREN, MED.AANT_AFD_UREN en MED.AANT_DNST_UREN berekend. De aandachtige lezer zal al gemerkt hebben dat er altijd gesommeerd wordt over het pad van een verwijzende sleutel heen. Het definiëren van een sommeringregel kan dus gezien worden als het toevoegen van een sommeringregels als kenmerk van een verwijzende sleutel. Als voorbeeld volgt de definitie van de sommeringregel voor kolom MED.AANT_AFD_UREN:

```
ALTER FOREIGN KEY FK_MED_MAFDTWS
(
        ADD CALCULATION RULE MED_AANT_AFD_UREN
        COMPUTE MED.AANT_AFD_UREN AS SUM OF ALL
                        RELATED MAFDTWS.AANT_UREN
);
```

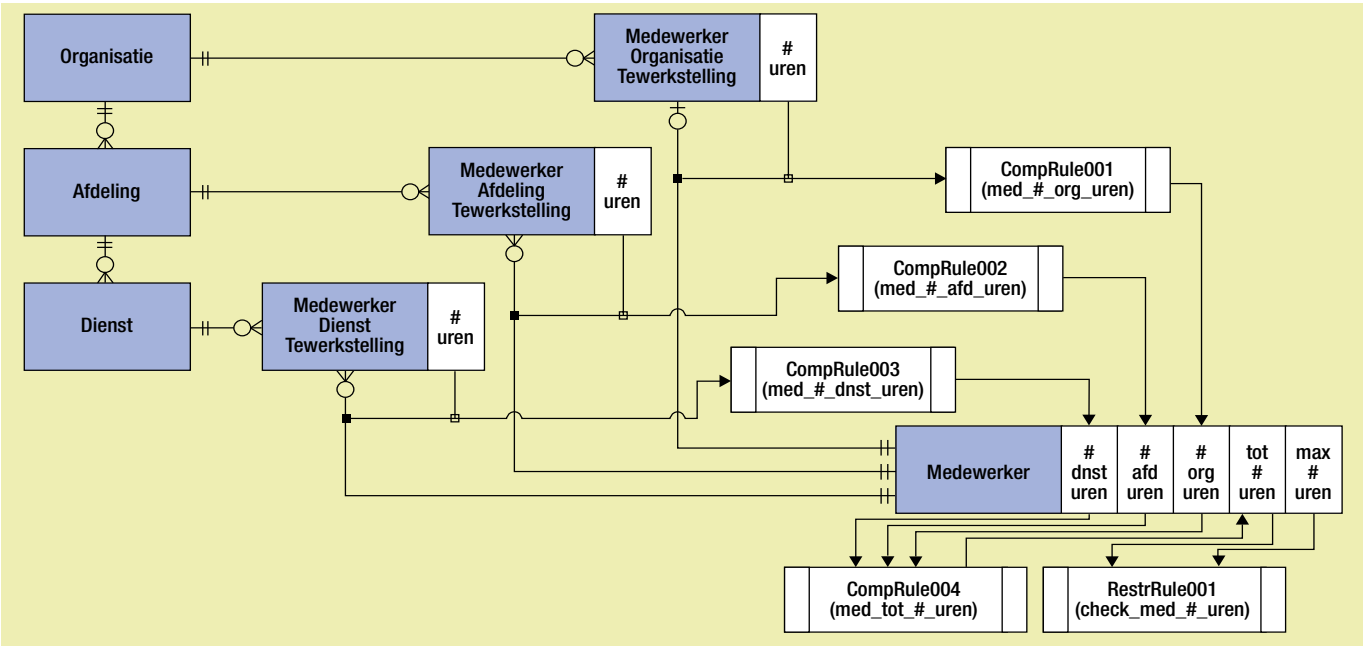
Een formule wordt gedefinieerd als een kenmerk van een tabel:

```
ALTER TABLE MEDEWERKER
        ADD FORMULA TOT_AANT_UREN
        COMPUTE AS      AANT_ORG_UREN + AANT_AFD_UREN
                                + AANT_DNST_UREN;
```

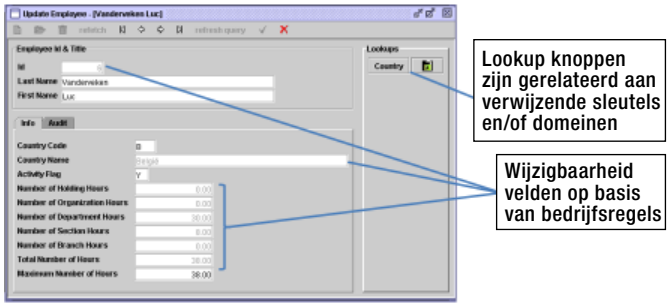
De beperkingregel die moet controleren of het totaal aantal uren dat een medewerker is toegewezen het maximum aantal uren dat afgesproken is met die medewerker niet overschrijdt, wordt gedefinieerd als gewone check constraint:

```
ALTER TABLE MEDEWERKER
        ADD CONSTRAINT
        CHECKMED.TOT_AANT_UREN <= MED.MAX_AANT_UREN;
```

Het gegevensmodel wordt uitgebreid met bedrijfsregels die opgenomen worden in de uitgebreide catalogus van de gegevensbank. Zo komen we tot het schema in afbeelding 1.
Een RDBMS dat in staat is zijn gepersisteerde gegevens consistent te houden met deze DDL-definities zou heel wat programmeerwerk overbodig maken. Dit houdt immers in dat zo'n RDBMS in staat zou zijn declaratief gedefinieerde kennisregels en beperkingregels af te dwingen. Minder behoefte aan procedurele code: zowel in triggers, als in stored procedures, als in applicatiecode. Het RDBMS is *rules capable* geworden en is zelf in staat regels uit te voeren en te controleren in de volgorde zoals aangegeven in het rules invocation diagram uit het vorige artikel. Dat diagram wordt evenwel overbodig indien het RDBMS rules capable is. Zo'n RDBMS bepaalt immers zelf wanneer regels uitgevoerd en/of gecontroleerd dienen te worden. Laten we het vanaf nu een Rules Capable Relational DataBase Management System (RCRDBMS) noemen.
Het onderhoud aan administratieve informatiesystemen kan sterk



Afbeelding 1: Rule-enriched gegevensmodel.



Afbeelding 2: Gebruikersinterface.

gereduceerd worden, mits er gebruik wordt gemaakt van zo'n RCRDBMS. Onderhoud aan administratieve informatiesystemen heeft in de meeste gevallen wijzigingen aan de gegevensstructuur tot gevolg. Met een bijkomende conversie van gegevens van de oude naar de nieuwe structuur er bovenop. Hieraan ontkomen we natuurlijk niet, maar een RCRDBMS laat niet toe zijn eigen regels te overtreden. Met een klassiek RDBMS is dat net zo. Maar een groot deel van de regels waaraan een administratief informatiesysteem dient te voldoen wordt over het algemeen op procedurele wijze geïmplementeerd, zodat het RDBMS bij een conversie niet in staat is deze regels te controleren. Wijzigingen aan de gegevensstructuur met bijhorende gegevensconversies houden enorme risico's in wat betreft de gegevensintegriteit. Arme applicatieontwikkelaars en DBA's.

We voegen een niveau departementen toe

Laten we even verder dromen en enig onderhoud doorvoeren, gebruikmakende van ons fictieve RCRDBMS. De organisatie is gestructureerd in afdelingen en diensten. Daarin komt op een bepaald moment verandering; er wordt immers beslist departementen toe te voegen aan deze organisatiestructuur. Departementen rapporteren aan de organisatie, terwijl afdelingen aan departementen rapporteren. Medewerkers kunnen vanaf nu ook tewerkgesteld worden op het niveau van departementen.

De volgende bedrijfsregel, hier uitgedrukt in 'mensentaal' blijft daarbij onveranderd: *Het aantal uren dat een medewerker tewerkgesteld wordt mag een tussen de personeelsdirecteur en die medewerker afgesproken maximum niet overschrijden.*

Wat er dient te gebeuren aan de gegevensstructuur is duidelijk:

1. Er dient een nieuwe tabel DEPARTEMENT aangemaakt te worden;
2. Er dient een verwijzende sleutel aangemaakt te worden van DEPARTEMENT naar ORGANISATIE;
3. De verwijzende sleutel van AFDELING naar ORGANISATIE dient verwijderd te worden;
4. De kolom AFD.ORG_ID is niet meer van toepassing en dient verwijderd te worden. In de plaats daarvan komt een nieuwe kolom AFD.DEP_ID;
5. Er dient een verwijzende sleutel aangemaakt te worden van AFDELING naar DEPARTEMENT;
6. Er dient een nieuwe tabel MED_DEP_TEWERKSTELLING gecreëerd te worden die verwijst naar de tabellen DEPARTEMENT en MEDEWERKER.

Elke DBA weet hoe dit moet. Met name het creëren van kolom AFD.DEP_ID en van de bijhorende verwijzende sleutel van AFDELING naar DEPARTEMENT stelt hierbij een probleem. Er dienen immers eerst departementen gecreëerd te worden waarnaar dan verwezen wordt vanuit afdelingen, alvorens het RDBMS toelaat deze kolom en deze verwijzende sleutel te creëren. Maar dat is gesneden koek voor DBA's.

De implementatie van de bedrijfsregel zelf vergt echter nog een aantal wijzigingen aan het gegevensmodel:

1. Er dient een virtuele kolom AANT_DEP_UREN toegevoegd te worden aan tabel MEDEWERKER:

```
ALTER TABLE MEDEWERKER
(
        ADD COLUMN AANT_DEP_UREN NUMBER VIRTUAL
);
```

2. Er dient een nieuwe kennisregel toegevoegd te worden die de berekening van deze kolom ondersteunt:

```
ALTER FOREIGN KEY FK_MED_MDEPTWS
(
        ADD CALCULATION RULE MED_AANT_DEP_UREN
        COMPUTE MED.AANT_DEP_UREN
            AS SUM OF ALL RELATED MDEPTWS.AANT_UREN
);
```

3. De kennisregel die kolom MED.TOT_AANT_UREN berekent, dient aangepast te worden:

```
ALTER TABLE MEDEWERKER
        DROP FORMULA TOT_AANT_UREN;
ALTER TABLE MEDEWERKER
        ADD FORMULA TOT_AANT_UREN
        COMPUTE AS  AANT_ORG_UREN + AANT_DEP_UREN +
                    AANT_AFD_UREN + AANT_DNST_UREN;
```

Alleen als de gegevens die gepersisteerd zijn in de gegevensbank voldoen aan al deze kennis- en beperkingregels slaagt men erin bovenstaand script uit te voeren. Het RCRDBMS zal er daarbij voor zorgen dat berekende kolommen netjes herberekend worden. Bovendien zal het systeem ervoor zorgen dat de gebruiker de berekende velden niet handmatig kan muteren: zo dient het statement UPDATE MEDEWERKER SET TOT_AANT_UREN = 30 te resulteren in een foutboodschap.

Dromen zijn bedrog

Maar een dergelijk RCRDBMS bestaat jammer genoeg niet. En aangezien relationele gegevensbanken door het merendeel van de ICT'ers gezien worden als ordinaire dozen gevuld met gegevens, hebben we niet al te veel te verwachten van de RDBMS-leveranciers. En daarbij blijft het niet. Tot nog toe hebben we alleen gekeken naar aspecten die te maken hebben met het afdwingen van bedrijfsregels binnen de gegevensbank zelf. Daarbij zijn we vergeten dat de presentatie van gegevens op gebruikerschermen ook bepaald wordt door bedrijfsregels. In sommige gevallen zelfs door dezelfde bedrijfsregels als de regels

die zorgen voor de gegevensconsistentie. Zo mag de gebruiker onder geen enkel beding over de mogelijkheid beschikken om de waarden van berekende kolommen te kunnen wijzigen. In afbeelding 2 wordt een gebruikersinterface weergegeven waarbij het de eindgebruiker onmogelijk gemaakt wordt de aantallen aan uren dat de medewerker is toegewezen aan de organisatie, departementen, afdelingen of diensten te wijzigen. Ook het totaal aantal uren dat de medewerker is tewerkgesteld is niet wijzigbaar via de gebruikersinterface. Hetzelfde geldt voor het id, aangezien ook die waarde bepaald wordt door het systeem. Zowat iedereen vindt het normaal dat dit presentatiegedrag geprogrammeerd wordt in de applicatiecode. Binnen de applicatie vindt men op diverse plaatsen applicatiecode terug die velden op *enabled* of *disabled* zet. De vraag is echter of het schrijven van zo'n applicatiecode wel normaal is. Bij gebrek aan een RCRDBMS valt deze codeerwoede nog te verdedigen.

Van droom naar werkelijkheid

Maar het kan ook anders. Zoals C.J. Date heeft beschreven dient men te beschikken over een soort van Virtual RDBMS. Rond de gegevensbank wordt een laag gelegd waarin al het fraais aan kennis- en beperkingregels op declaratieve wijze gedefinieerd wordt. Bovendien zorgt deze laag ervoor dat de gegevens die gepersisteerd zijn in de gegevensbank te allen tijde voldoen aan deze regels. En dat de applicatiesoftware die gebruikmaakt van deze gegevensbank in staat is haar presentatieaspecten op te vragen via deze laag. Het komt er dan op neer de gebruikers-interface zo te schrijven dat deze automatisch reageert op de kennis- en bedrijfsregels. Dergelijke systemen bestaan, in de vorm van data-centric rules engines. En Barbara von Halle vernoemt zelfs twee van zulke engines; ze geeft daarbij tekst en uitleg over hoe bedrijfsregels geïmplementeerd worden binnen dergelijk systeem. Vanwege enige vorm van sluikreclame laat ik het aan de lezer van dit artikel over om uit te pluizen om welke leveranciers het gaat. Wel kan ik vertellen dat één van beide spelers zijn systeem op Nederlandse bodem ontwikkelt en daarbij een pertinente rol speelt op de Nederlandse markt.
Bij gebruikmaking van een dergelijk product vergen presentatieaspecten zoals de wijzigbaarheid van velden, lookup-knoppen en zo meer geen codeerwerk meer. Al deze zaken worden op declaratieve wijze ingeregeld door middel van presentatieregels, die als speciaal geval van kennis- en bedrijfsregels gezien kunnen worden. Overigens is het voorbeeldscherm in afbeelding 2 gebouwd op basis van een zelfgebouwd RCRDBMS dat moet dienen als *proof of concept*. De zelfbouw en het productierijp maken van zo'n RCRDBMS vergt echter een gigantische investering, die de aanschaf van een commercieel product meerdere malen overschrijdt.

Conclusie

In dit artikel is een fictief systeem beschreven dat gezien kan worden als een RDBMS waaraan enkele functionaliteiten toegevoegd worden die het RDBMS rules capable maken. Zo wordt het verheven tot een RCRDBMS. Bij administratieve informatiesystemen is de consistentie van de gegevens van groot belang. Een RCRDBMS zorgt ervoor dat de graad van consistentie van de gepersisteerde gegevens vele malen groter is dan mogelijk is door middel van een standaard RDBMS. Dat geldt zowel bij de initiële bouw van administratieve informatiesystemen als bij het onderhoud ervan.
Jammer genoeg bestaat een dergelijk RCRDBMS niet, en kunnen we van de RDBMS-leveranciers ook niet al te veel verwachten. Data-centric rules engines kunnen echter wel soelaas brengen, aangezien ze een laag vormen om het RDBMS heen. Voor de gebruiker van het RDBMS lijkt het dan alsof hij beschikt over een RCDBMS. Bovendien voldoet deze architectuur beter aan een eis die opgelegd wordt door de Business Rules Approach, namelijk die van scheiding van gegevens en bedrijfsregels. Een beschrijving van hoe zo'n RCRDBMS eruit kan zien wordt gegeven door C.J. Date[3]. Barbara von Halle onderkent het bestaan van dergelijke systemen en legt zelfs uit hoe bedrijfs-regels geïmplementeerd worden in zo'n systeem[4].
Zoals aangegeven in het vorige artikel worden bedrijfsregels in de vorm van complexe volzinnen omgezet in kleinere samenhangende kennisregels en beperkingregels. Elk van deze regels wordt gedefinieerd binnen het RCRDBMS. Op welke informatie de regels inspelen en wanneer, valt dan af te leiden uit het zo ontstane rule-enriched gegevensmodel. Bij gebruikmaking van een RCRDBMS wordt voldaan aan drie belangrijke aspecten die de Business Rules Approach oplegt, namelijk het begrijpelijk maken van de bedrijfsvoering voor de eindgebruiker, de traceerbaarheid van waar en wanneer bedrijfsregels uitgevoerd en/of gecontroleerd worden, en de scheiding van gegevens, processen en bedrijfsregels. Bovendien zijn deze kleine samenhangende en samenwerkende regels gemakkelijker onderhoudbaar dan grote complexe volzinnen. Men komt gemakkelijker tot hergebruik van regels, niet in het minst aangezien deze eenvoudige bedrijfsregels op een declaratieve wijze gedefinieerd kunnen worden. Als we de aanhangers mogen geloven, zou deze aanpak moeten zorgen voor informatiesystemen die robuuster en flexibeler zijn en minder onderhoud vragen, en die beter voldoen aan de behoeften 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 bedrijfs-regels 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.*
3. *C.J. Date, What Not How: The Business Rules Approach to Application Development.*
4. *Barbara von Halle, Business Rules Applied, Building Better Systems Using the Business Rules Approach.*

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