CDM RuleFrame bewijst meerwaarde

Business rules ontwerpen en modelleren

De ontwikkelingen in de Oracle-wereld gaan snel. Java, SOA en Application Express zijn allemaal nieuwe en interessante producten. In de praktijk wordt in een groot deel (misschien wel het grootste deel) van de Oracle ontwikkeltrajecten echter nog steeds gebruik gemaakt van wat we 'traditionele' Oracle-ontwikkeltools noemen. Het ontwerpen en modelleren van bedrijfsregels ofwel business rules, speelt binnen die trajecten een belangrijke rol.

Het CDM RuleFrame is een aantal jaren geleden door Oracle Consulting Nederland ontwikkeld voor het implementeren van business rules. Het is een onderdeel van Headstart (niet te verwarren met JHeadstart) en biedt een krachtig en compleet framework om business rules te ontwerpen en implementeren op basis van table triggers en/of view instead-of-triggers, met behulp van Oracle Designer en PL/SQL.
Designer biedt zelf ook mogelijkheden om business rules te implementeren, maar geen eenduidige werkwijze om business rules te registreren en genereren. Het gevolg is dat iedere programmeur zijn eigen methoden kan hanteren wat het onderhoud niet bepaald eenvoudiger maakt.
Dit artikel geeft een indruk van de manier waarop het ontwerp van business rules tot stand komt en de implementatiewijze met het CDM RuleFrame. Het gaat dieper in op de voor- en nadelen van het gebruik van CDM RuleFrame in een Oracle Designer-omgeving.

Business rules

In de eerste plaats is een eenduidig begrip van een business rule van groot belang. De vraag is dus: wat is een business rule volgens CDM? Een business rule beschrijft een beperking op data of een automatische actie als gevolg van datamanipulatie. De business rule logica levert bij datamanipulatie een goed- of fout-situatie op. Het kan ook een andere datamanipulatie of een applicatiefunctie activeren. Een business rule wordt gevalideerd bij het selecteren, invoeren, wijzigen of verwijderen van een item of record. Dit wordt het event genoemd. Een event kan ook een schedule zijn, bijvoorbeeld 'elke maandag om 03:00u'.

Ontwerp

Veel moderne applicaties kennen minimaal drie 'lagen' (ook wel *tiers* genoemd): het RDBMS, middleware en een cliënt. Een tier is een zelfstandig onderdeel van een distributed system. Een applicatie die maar uit één module bestaat, en dus niet gedistribueerd is, bestaat maar uit één tier. Internet wordt meestal gebruikt met twee programma's, een webbrowser en een webserver: dit wordt dus een 2-tier systeem genoemd. Wanneer de webserver gegevens uit een database haalt wordt dit 3-tier. Business rules kunnen in theorie binnen elke tier worden geïmplementeerd. Als in Oracle Designer het CDM RuleFrame wordt ingezet, worden de business rules automatisch in het RDBMS geplaatst. Dat komt omdat de implementatie van de business rules in PL/SQL plaatsvindt. Bij het ontwerp worden rules gecategoriseerd, de zogenaamde business rule-classificatie. In de volgende paragrafen worden de verschillende categorieën besproken.

Randvoorwaarden	Type	Voorbeeld
Slechts 1 attribuut is betrokken	ATT - Attribute Rule	Het salaris is altijd een positief geheel getal
Alleen attributen uit dezelfde rij zijn betrokken	TPL - Tuple Rule	Einddatum project is altijd groter dan begindatum project
Attributen uit meerdere rijen binnen een entiteit zijn betrokken	ENT - Entity Rule	Er zijn niet meer dan 15 afdelingen toegestaan
Het gaat om attributen uit meerdere entiteiten	IER – Inter Entity Rule	De begin- en einddatum van een projectdeelname moet liggen tussen de begin- en einddatum van het project

Tabel 1.

Randvoorwaarden	Type	Voorbeeld
Er geldt een verplichting bij het aanmaken van een nieuw record	CRE – Create Rule	Een nieuwe taak kan
		alleen toegekend worden aan een werknemer die in dienst is
Aan deze voorwaarde moet voldaan worden wanneer er een update plaatsvindt	UPD – Update Rule	Men kan alleen opslag geven aan een werknemer die in dienst is
Deze voorwaarde moet gelden bij een insert, update of delete	MOD – Modify Rule	Een werknemer kan alleen geplaatst worden op of verplaatst worden naar een geldige afdeling
Geldig bij een delete	DEL – Delete Rule	Een werknemersdossier kan
		alleen gewist worden wanneer deze uit dienst is

Tabel 2.

Static rules

We spreken van een *static rule* als de data te allen tijde gevalideerd kunnen worden tegen de gestelde beperking. De validatie hoeft dus niet op moment van datamanipulatie te zijn, maar kan ook in een later stadium plaatsvinden, bijvoorbeeld met behulp van een controle query. Voorbeelden van static rules zijn "het salaris is altijd een positief geheel getal" of "de begindatum moet voor de einddatum liggen" of "de som van alle projectbudgetten mag het afdelingsbudget niet overschrijden". Static rules zijn vaak de meest voorkomende rules. Binnen deze categorie onderscheidt men de types zoals genoemd in tabel 1.

Dynamic rules

Dynamic rules kunnen slechts tijdens de datamanipulatie gecontroleerd worden. Veelal is dat omdat de oude en de nieuwe waarde nodig zijn om de rule te valideren, maar het kan ook omdat de systeemdatum een rol speelt bij de validatie. Bijvoorbeeld "de mogelijke overgangen van burgerlijke staat zijn, alleenstaand - getrouwd, getrouwd - gescheiden, gescheiden - getrouwd, getrouwd - weduwe, weduwe - getrouwd" of "het salaris van een medewerker mag alleen gewijzigd worden als de datum 'uit dienst' nog niet is verstreken". Ook hier geldt een onderverdeling (zie tabel 2).

Type	Voorbeeld
CEV – Change Event Rule	Wanneer een contract gemuteerd wordt, schrijf en record weg naar contract mutaties (audit)
DFT – Complex Change Event Rule	Wanneer er een nieuwe werknemer aangemaakt wordt, ken deze default de afdeling van de aangelogde gebruiker toe

Tabel 3.

DML change event rules

Bij *DML change event rules* triggert de data manipulatie een andere data manipulatie. Het vullen van een audit tabel of bijwerken van een redundante kolom valt binnen deze categorie, maar ook het toekennen van default waarden (zie tabel 3).

Non-DML change event rules

Bij *non-DML change event rules* wordt de datamanipulatie gevolgd door een actie waarbij geen data worden gemanipuleerd. Het gaat in dit geval om bijvoorbeeld het versturen van een e-mail of het printen van een rapport. Dit zijn applicatiefuncties buiten de database die niet (door de database) terug te draaien zijn (zie tabel 4)

Type	Voorbeeld
CEW – Change Event without DML	Bij indiensttreding, stuur een nieuwe werknemer een welkomstbrief

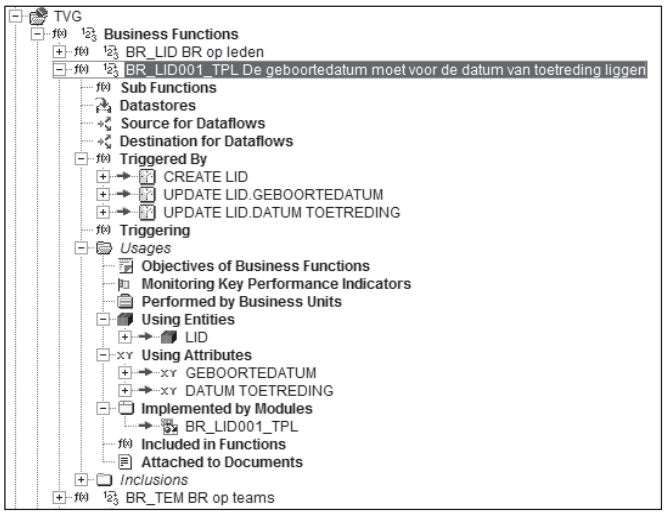
Tabel 4.

Authorization rules

In het geval van *authorization rules* kan men denken aan autorisatie door middel van rolverdeling binnen een applicatie. Een manager is bijvoorbeeld de enige die de salarissen mag wijzigen, maar ook de enige die door middel van een rol toegang heeft tot het scherm waarin dit mogelijk is.
In Oracle Designer kan de business rule als zogenaamde Business Function worden vastgelegd waarna deze met behulp van het CDM RuleFrame kan worden geïmplementeerd. De omschrijving van deze Business Function is van belang, omdat deze later de foutmelding wordt bij overtreden van de rule. Na het koppelen van de nodige Events en Entities aan de Business Function is het functioneel ontwerp van de business rule compleet.

Implementatie

CDM RuleFrame stelt een standaardwerkwijze voor de implementatie voor. Daarbij is de categorisering en de gedetailleerde typering van de rules belangrijk. Elk type rule biedt een eenduidige implementatiewijze.
Een groot aantal van de types binnen de categorieën *static rules* en *dynamic rules* kunnen geïmplementeerd worden als 'native property'. Daarmee wordt bedoeld dat er al een implementatiewijze in de database voorhanden is of dat de standaard Table API van Oracle Designer de implementatie voor z'n rekening neemt. Voor deze business rules wordt de CDM RuleFrame implementatie dus niet gebruikt. Bij de implementatie van zo'n 'native property' valt te denken aan het gebruik van datatypes, primary keys, unique keys en foreign keys. Ook de meeste *authorization rules* kunnen eenvoudig geïmplementeerd worden als



Afbeelding 1. Business Function in Designer.

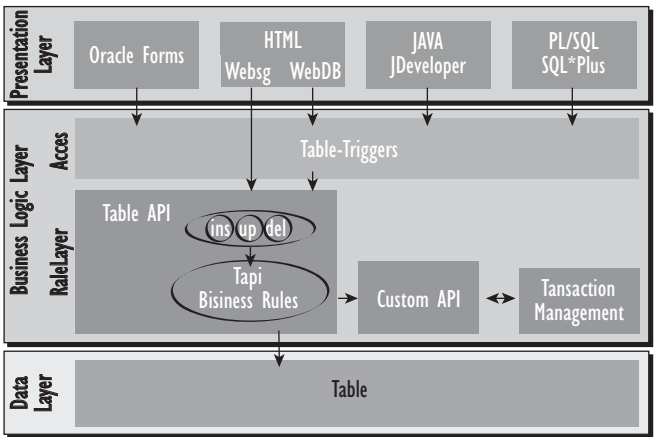
'native property'. De database biedt daarvoor database authentication, roles, table privileges en de virtual private database. De Table API verzorgt ook een aantal 'native property's'. De TAPI is een standaard te genereren package van Oracle Designer. Elke tabel heeft een TAPI met daarin zogenaamde table handlers om DML-operaties op deze tabel uit te voeren, ofwel: PL/SQL-procedures ten behoeve van een insert, update, delete of lock. De TAPI verzorgt onder andere default values, het afdwingen van UPPERCASE voor een kolomwaarde en een arc-implementatie.
Aanvullend op al deze 'native property's' worden business rules geïmplementeerd met het CDM RuleFrame. Overigens kunnen vrijwel alle 'native property'-implementaties ook in het CDM RuleFrame worden geïmplementeerd, maar dat betekent al snel een aanzienlijke verslechtering van de performance.
De architectuur van het CDM RuleFrame wordt getoond in het onderstaande schema. Via database triggers of view instead-of-triggers wordt de Table API en vervolgens de Custom API van de betreffende tabel aangeroepen. Deze CAPI vormt samen met het standaard transaction management package het hart van het CDM RuleFrame.

De CAPI bevat voor elke business rule één zogenaamde 'need'-functie, die controleert of een rule als gevolg van een DML event gevalideerd moet worden. Daarnaast bevat de CAPI per business-rule één validatie-functie, die per DML-event goed of fout oplevert. Na het ontwerpen van een business rule zoals hierboven beschreven, moet de programmeur slechts de volgende stappen uitvoeren voor de implementatie van een business-rule in de CAPI:

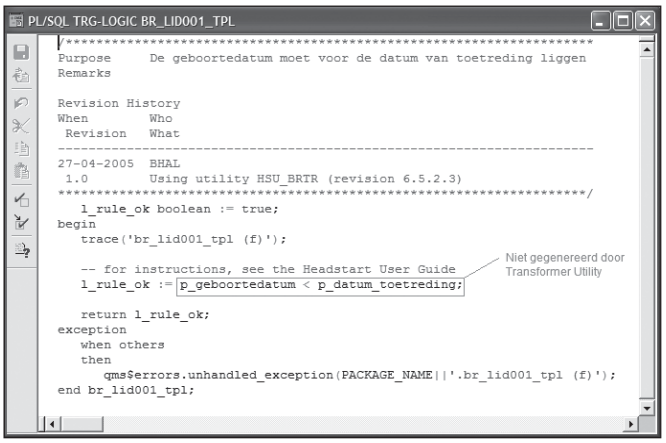
- Start de Business Rule Design Transformer utility;
- Codeer de validatie-functie;
- Start de Bundle CAPI utility;
- Genereer de CAPI-package en eenmalig de TAPI package en TAPI triggers.

Het coderen van de validatie-functie is het enige programmeerwerk dat noodzakelijk is. De Business Rule Design Transformer utility en de Bundle CAPI utility zorgen ervoor dat dit programmeerwerk tot een minimum beperkt blijft. In onderstaand voorbeeld is te zien wat er in de Custom API door de programmeur is gecodeerd.

Een belangrijke eigenschap van het CDM RuleFrame is het transactie-mechanisme. Zie daarvoor onderstaande SQL*Plus sessie. Na het sluiten van de transactie blijkt dat er fouten zijn gemaakt. Om deze fouten te achterhalen wordt het script gebruikt dat de fouten stack uitleest. Overigens ontstaat er ook een fout als de transactie niet via de transaction management package wordt afgesloten maar direct via commit:



Afbeelding 2. Schema van een logische 3-tier.



Afbeelding 3. Validatie-functie van de business rule.

```
SQL> exec qms_transaction_mgt.open_transaction(user)

PL/SQL procedure successfully completed.

SQL> update tvg_leden
 set geslacht = 'X'
 where nummer = 100;

1 row updated.

SQL> update tvg_leden
 set datum_toetreding = to_date('01-JAN-1900', 'DD-MON-YYYY')
 where nummer = 112;

1 row updated.

SQL> exec qms_transaction_mgt.close_transaction(user)

ORA-20998: Transaction Failed
   ORA-06512: at "HST65.QMS$ERRORS", line 128
   ORA-06512: at "HST65.QMS_TRANSACTION_MGT", line 840

SQL> @messages.sql

Error TVG-00020: 112: De geboortedatum moet voor de
datum van toetreding liggen.
   Error TVG-00018: 100: Geslacht moet M of V zijn.
```

De front-end moet dus rekening houden met fout ORA-20998 en zal bij deze fout de fouten stack moeten uitlezen. Het form template van Headstart is daar al geschikt voor gemaakt, zoals in onderstaande schermprint te zien is.

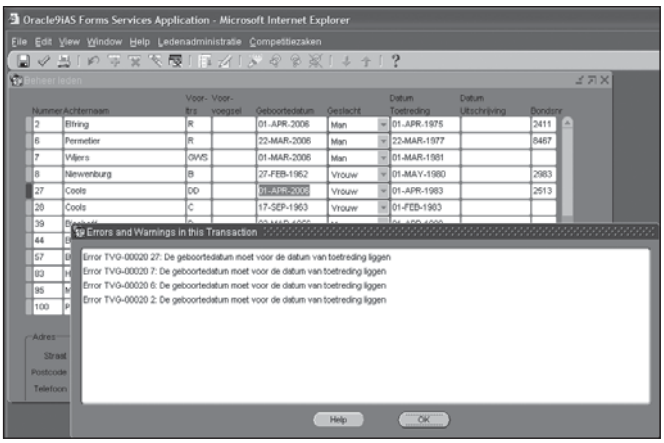
Meerwaarde

Het CDM RuleFrame bewijst zijn meerwaarde op verschillende punten. Een aantal belangrijke voordelen zijn:

- Het CDM RuleFrame voert zijn controles uit op transactieniveau. De gebruiker kan dus zelf bepalen bij welk DML-statement een transactie start en bij welk statement de transactie eindigt, oftewel wanneer er gevalideerd dient te worden. Hierbij worden alle gedefinieerde business rules gecontroleerd, voordat de meldingen getoond worden. Dit heeft als voordeel dat de gebruiker niet na het herstel van iedere fout weer moet controleren of er misschien nog een fout gemaakt is. Daarnaast kunnen controles die voorheen lastig in de database afgevangen konden worden en daarom in de cliënt applicatie opgenomen waren, nu ook in de database afgevangen worden. Bijvoorbeeld: "Het totaal bijboekingen in kasboeksaldo moet gelijk zijn aan alle bijboekingen in kasboekregels". Indien beide tabellen in een willekeurige volgorde gewijzigd worden in dezelfde transactie, kan deze controle pas gevalideerd worden nadat de complete transactie is doorgevoerd. De controle kan daardoor niet op één van de tabellen geplaatst worden. Door het transactiemechanisme dat het CDM RuleFrame in zich heeft, kan deze controle wel door het CDM RuleFrame in de database uitgevoerd worden.
- Omdat de logica in de database gecodeerd wordt kan men onafhankelijk hiervan met verschillende cliënt applicaties werken, bijvoorbeeld een eenvoudige HTML applicatie voor extern gebruik op het Internet en een zwaardere op Oracle Forms gebaseerde applicatie voor interne medewerkers.
- Het aantal of de complexiteit van de business rules kan uitgebreid worden zonder dat de applicatie aangepast hoeft te worden. Hierdoor kunnen ontwikkel- en opleidingskosten gespaard worden.

Kan het nog beter?

Het CDM RuleFrame laat weinig te wensen over. Performance is wel een issue bij het CDM RuleFrame, maar sinds patch Headstart 6.5.2.3 is dat al behoorlijk verbeterd, getuige ook een zinsnede van de readme.txt van deze patch:
"During development and testing of this patch, the following improvements of the CDM RuleFrame engine were measured on an Windows 2000 laptop with RDBMS 8.1.7.1.1. 1000 DML statements, 3 Business Rules per statement performance improved from 78.79 sec to 19.260 sec (76%)".
Dat zijn nog steeds geen geweldige rapportcijfers maar er bestaat ook de mogelijkheid om een business rule of alle business rules van een tabel (in de huidige sessie) tijdelijk uit te zetten. Ook bevat de CAPI een procedure om achteraf alle statische rules te valideren. Dat kan natuurlijk van pas komen bij batchverwerking.
Ondanks de hier genoemde voordelen wordt het CDM RuleFrame in de praktijk betrekkelijk weinig gebruikt. Dat heeft enerzijds vooral te maken met de afhankelijkheid van Oracle Designer. Zonder Oracle Designer is een CDM RuleFrame implementatie zoals hierboven beschreven niet mogelijk. Anderzijds heeft het ook te maken met een gebrek aan kennis van de implementatie. Zo lijkt de implementatie van het CDM RuleFrame voor een leek op een trukendoos, want er worden nogal wat code-generatoren gebruikt. Dat is erg jammer, want de ontwikkelsnelheid is bijzonder hoog. De meeste inspanning



Afbeelding 4. Uitlezen van de fouten stack door Forms.

zit 'm in het ontwerp. Vervolgens kan (met enige begeleiding) elke willekeurige PL/SQL-ontwikkelaar business rules implementeren met het CDM RuleFrame.
De onzekere toekomst van Oracle Designer kan gelukkig enigszins worden gerelativeerd omdat juist in Designer het beheer van het datamodel van Oracle Applications plaatsvindt. Ook is op dit moment nog steeds een groot deel van de front-end van Oracle Applications gebouwd in Forms en Reports. Het lijkt erop dat Designer, Forms en Reports alleen al om die reden voorlopig nog een lange tijd zullen worden ondersteund. Designer heeft nog steeds een hele hoge productiviteit, zeker in combinatie met de iDevelopment Accelerators (Headstart en CDM RuleFrame). Het is een volwassen, volwaardig, uitontwikkeld en dus zeer stabiel product, met bewezen technologie.

Referenties

- CDM Ruleframe – The business rule implementation framework that saves you work, Oracle Corporation, iDevelopment Center of Excellence
- Custom Development Method Classic 2.6
- Headstart Oracle Designer User guide 6.5.3.0
- http://www.oracle.com/technology/consulting/9iServices/idev.html
- http://www.anewlink.nl/publicaties/opt0009tekst.htm

Job Heisen en **Martin Schapendonk** zijn consultants bij Whitehorses. Dit artikel is gebaseerd op een eerder gepubliceerd whitebook van Bas van Hal (www.whitehorses.nl/whitebooks).

NIEUWS

Artikelen met praktische informatie, geschreven door en bestemd voor Oracle-professionals vindt u in het Online Archief van Array Publications. Vaktijdschriften als Database Magazine, Software Release en Java Magazine hebben hun artikelenarchief online gezet. Met een heldere zoekstructuur vindt u snel wat u zoekt op www.optimize.nl.

Atos Origin biedt databaseservice op basis van Oracle 10g Grid-technologie

Atos Origin en Oracle bieden een nieuw alternatief voor klanten die zoeken naar schaalbare databasecapaciteit, in combinatie met hoge betrouwbaarheid en betaling op basis van werkelijk gebruik. Utility Based Grid is een hosted versie van Oracle 10g Grid die de technologische kracht van Oracle combineert met de hostingervaring van Atos Origin en een flexibel afrekenmodel.
In antwoord op de groeiende vraag naar zeer snel schaalbare, maar betrouwbare informatiesystemen - bijvoorbeeld voor telecommunicatie, service websites of real-time omgevingen - lanceert Atos Origin Utility Based Grid, dat is gebaseerd op het Oracle-10g-databaseplatform. De dienstverlening, die tegen zeer scherpe tarieven wordt aangeboden, omvat databasehosting en optioneel ook flexibele licenties, met afrekening naar werkelijk gebruik. In de dienstverlening kunnen ook webservices en applicaties opgenomen worden.
"Atos Origin heeft in nauwe samenwerking met Oracle het Utility Based Grid ontwikkeld. Met ons technische team in de Verenigde Staten zijn aanpassingen uitgewerkt, die gedetailleerde rapportages over het gebruik per klant van een Oracle 10g Grid-database mogelijk maken", zegt Ron Augustus, Director Technology Solutions Oracle Benelux. "Voor organisaties is het uiteraard belangrijk dat het Grid-concept voor databases zich duidelijk in de praktijk bewezen heeft en dat RAC - de onderliggende technologie van 10G Grid die high-availablity mogelijk maakt - al in productie draait bij meer dan 1200 klanten van ons."

Oracle koopt SPL

Oracle neemt de SPL Worldgroup over, een leverancier van beheersoftware voor de bedrijfsvoering en opbrengstenstroom van nutsbedrijven en software voor belastingmanagement bij overheidsinstanties. Charles Phillips, President van Oracle: "Met de toevoeging van SPL aan ons portfolio zijn we van plan een volledige end-to-end oplossing voor bedrijfsvoering en opbrengsten voor leveranciers van openbare voorzieningen op de markt te brengen. Het maakt daarbij niet uit of de voorzieningen geprivatiseerd zijn of niet. Traditionele maatwerkoplossingen zijn inefficiënt, duur en niet flexibel gebleken. Oracle biedt deze leveranciers meer inzicht in hun bedrijfsvoering, zodat ze meer winsten kunnen halen uit diensten en hun klantentrouw kunnen verhogen. Daarnaast voldoet de belastingbeheeroplossing van SPL aan de behoeften van overheidsorganisaties voor beheer van hun inkomsten."

Oracle neemt MetaSolv Software over

Oracle neemt MetaSolv Software over, een bedrijf dat ondersteunende diensten levert voor de telecommunicatie en mediabranche. MetaSolv Software wordt overgenomen door een aandelenovername van 4,10 dollar per aandeel, wat een overnamebedrag oplevert van ongeveer 219,2 miljoen dollar. De overname moet nog worden goedgekeurd, onder meer door de aandeelhouders. Naar verwachting zal dit eind 2006 of begin 2007 worden afgerond.