ABIS NIEUWSBRIEF VOOR DB2 PROFESSIONALS

JAARGANG 4 - NUMMER 3, APRIL 2006

OPEN CURSOR

Data Base Administrator, het zal je job maar wezen.
In lang vervlogen tijden was je verantwoordelijk voor het maken, onderhouden en tunen van databases. Enkel grondig gecontroleerde SQL requests (lees static SQL) werden toegelaten. Ondertussen weten we wel beter. De creativiteit van de applicatieontwikkelaars is sterk toegenomen. Java, JDBC, dynamic SQL, het moet allemaal kunnen. Snel en nauwelijks gecontroleerd, die indruk krijg je althans. Maar wat wil je, ieder zijn job. Jij de databases, en 'zij' de SQL.
Maar daar blijft het niet bij. Ontwikkelingstools gaan meer en meer ondersteuning bieden voor het definiëren van databases - lees het eerste artikel in dit nummer maar eens. Nu nog vooral bedoeld om te testen en te experimenteren, maar dat kan snel veranderen. Blijf op de hoogte!

Het ABIS DB2-team.

IN DIT NUMMER:

CLOSE CURSOR

Volgende keer sluiten we de serie over de DB2 Content Manager af met een artikel over hoe men het systeem kan gebruiken.

Ook de serie over gelijkenissen en verschillen tussen DB2, Oracle en SQLServer krijgt een vervolg.

Tot dan!

RAD en DB2

Diane Hendrix (ABIS)

Wat is RAD?

Rational Application Developer (RAD), de opvolger van WebSphere Studio Application Developer (WSAD) is een krachtige IDE (Integrated Development Environment) gebaseerd op Eclipse. RAD maakt deel uit van het Rational Software Development Platform, de familie van IBM ontwikkelingstools waartoe ook WebSphere Developer for z/Series (WD4Z), het vroegere WebSphere Studio Enterprise Developer (WSED), behoort. Deze tools zijn ontworpen om gebruikt te worden in een grote verscheidenheid van ontwikkelingsrollen zoals die van Java-ontwikkelaar, webontwikkelaar, enterprise applicatieprogrammeur, business analist en systeemarchitect. Natuurlijk mag in dit plaatje ook de mogelijkheid om rechtstreeks via het tool met een database server te communiceren niet ontbreken. Het gaat echter verder dan enkel communicatie! Zowel een DBA als een ontwikkelaar kunnen RAD gebruiken om een aantal van hun database manipulaties te verrichten, onder meer ook manipulatie en configuratie van allerhande database objecten. RAD voorziet in het gebruik van een brede waaier van database servers. Naast de meegeleverde Cloudscape database, worden ook bijna alle relationele databases ondersteund (DB2, Oracle, SQL Server, Sybase, Uniforme) en dit voor verschillende versies en platformen. Aan mogelijkheden dus geen gebrek! In dit artikel richten we ons uitsluitend op de mogelijkheden die RAD biedt voor DB2 UDB V8.2. Voor andere databases/versies/platformen zijn de overeenkomsten echter erg groot. De ontwikkeling van database applicaties met RAD, zal aan bod komen in een later nummer van Exploring DB2.

RAD terminologie

De IDE van RAD wordt de Werkbank genoemd. In deze Werkbank zal de ontwikkelaar zijn gepersonaliseerd ontwikkelomgeving samenstellen aan de hand van een aantal RAD perspectieve. Een RAD perspectieve is in feite niet meer dan een verzameling van editors en views die vereist zijn voor een bepaalde ontwikkelaarsrol. Zo is er bijvoorbeeld een Java perspective voor de ontwikkeling van Java-applicaties, een Web perspective voor de ontwikkeling van webapplicaties en natuurlijk ook een Data perspective. Via de verschillende editors en views aanwezig in het Data perspective kan de DBA/ontwikkelaar datadefinities aanmaken en manipuleren. Alle informatie, instellingen, coding en ook datadefinities worden gegroepeerd in een RAD- project. Zo zijn er bijvoorbeeld Java-projecten, webprojecten,....

Database-definities en RAD

RAD gebruikt het XML Meditatie Interchange (XMI) formaat om zijn database-definities op te slaan en uit te wisselen. Voor elk database-object dat in RAD aangemaakt of binnengebracht wordt, bestaat er een lokale XMI descriptor. RAD voegt er voor elk type van object een prefix aan toe, zodat er onder meer .dbxmi (database), .tblxmi (ta-

bel) en .schxmi (schema) bestaan. De inhoud van deze XML files kan bekeken en gewijzigd worden door gebruik te maken van gespecialiseerde editors. Ook kan, indien nodig, van elke XMI file een SQL DDL script gegenereerd worden.

Welke DB2-acties zijn er allemaal mogelijk in RAD?

In het Data perspective van RAD kan de database DBA/ontwikkelaar o.m. de volgende taken verrichten.

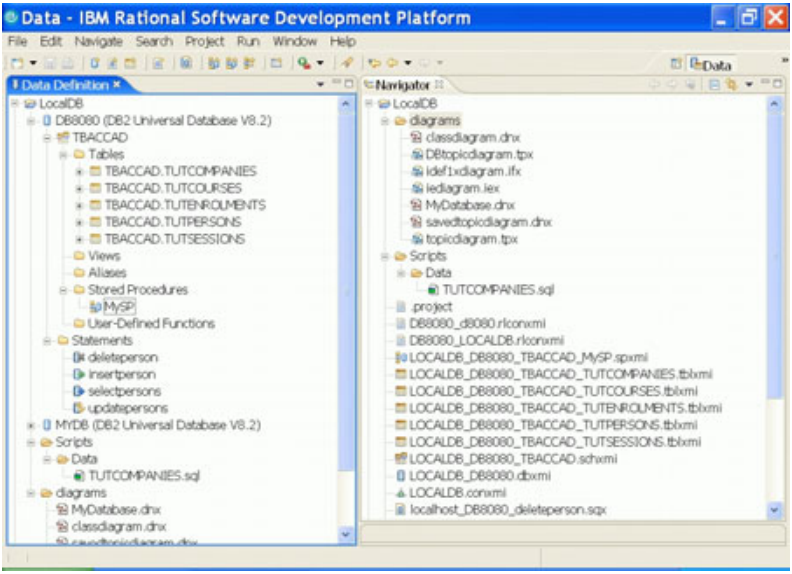
- aanmaken van een database connectie;
- importeren van database-definities in een project;
- wijzigen en deployen van geïmporteerde database-definities;
- creëren en uitvoeren van SQL DDL en DML scripts;
- genereren van XML (schema, DTD, ...) vertrekkende van allerlei database-definities;
- aanmaken van database-objecten gebruik makende van een logisch modelleerdiagram. Hiervoor zijn verschillende types van diagrammen voorzien. Omgekeerd kunnen ook diagrammen gegenereerd worden vertrekkende vanaf bestaande database-definities;
- creëren, uitvoeren en debuggen van Stored Procedures en User Defined Functions.

Op dit moment is er in RAD nog geen ondersteuning voor het modelleren van indexen, check en unieke constraints, triggers, structured types en identity-kolommen.

Aanmaken van een connectie met een bestaande database

Voor het aanmaken van een connectie gebruikt men de New Database Connection wizard in het Database Explorer view. Zowel lokale als remote databases kunnen bereikt worden door gebruik te maken van de standaard JDBC drivers (type 2 of 4). Deze connecties kunnen indien nodig ten allen tijde gewijzigd, gerefreshed, gesloten of geopend worden. Ook kan vanuit dit view de database-inhoud opgevraagd worden. Een voorbeeld van het Data perspective is te zien in Figuur 1. In het DB Explorer view zie je een connectie naar een lokale DB2 V8.2 database server. Hier vind je niet enkel de logische naam en type van de connectie, maar ook de aanwezige schema's, tabellen, views, User Defined Functions (UDFs) en Stored Procedures (SPs). Voor elke kolom wordt meegegeven van welk datatype ze is, en of er referentiële constraints bestaan. In het DB Output view zie je de inhoud van één van de tabellen. Het onderliggende SQL SELECT statement en eventuele problemen/fouten bij de uitvoering hiervan bevinden zich in onderliggende panels van dit view (niet getoond). De SQL editor toont een SQL DDL script dat gegenereerd werd op basis van een bestaande tabeldefinitie. Indien er geconnecteerd wordt met een federated database server, kan je de federated tables (nicknames) terugvinden in de Aliases folder.

Figuur 1: het Data perspective in RAD

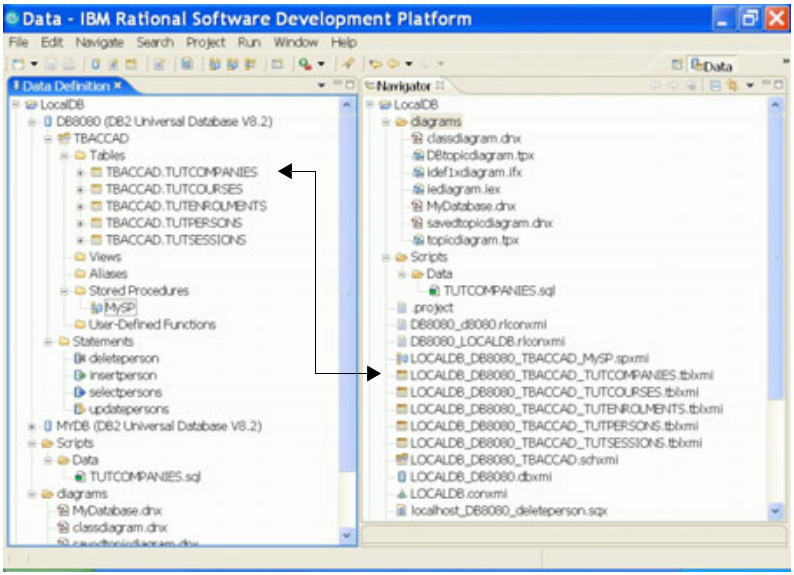


Importeren van database-definities in een project

Het Database Explorer view is read-only. Als men de bestaande definities wil wijzigen of nieuwe wil aanmaken, moet men de database-definities importeren in een RAD-project. In Figuur 1 zie je links boven het Data Definition view met een aantal database- projecten.

In Figuur 2 zie je van dit view een uitvergroting, samen met het Navigator view. Dit laatste view toont van alle database objecten het .xmi bestand. Databases, schema's en tabellen van een project worden in beide views met duidelijke pictogrammen aangeduid. Als je via de wizards SQL DDL scripts genereert (zie verder) komen die per default terecht in de Scripts folder. Voor SQL DML files is dit de Statements folder.

Figuur 2: het Data Definition en Navigator view in RAD



Aanmaken en uitvoeren van SQL DDL- en DML-scripts

Scripts voor de creatie van alle database-objecten kunnen via de RAD workbench aangemaakt worden. Hiervoor kan je de voorziene wizards gebruiken, ofwel alles volledig zelf schrijven in de SQL-editor (voorzien van syntax highlighting, syntax check en content assist). Er zijn wizards voorzien voor de creatie van databases, schema's, tabellen en views. De DDL-scripts kunnen vervolgens rechtstreeks op de database server naar keuze uitgevoerd worden (deploy). Er kan gebruik gemaakt worden van een bestaande database connectie, of men kan een nieuwe connectie definiëren. Enkel het script voor de creatie van een database kan niet vanuit de RAD Workbench gedeployed worden. Voor het aanmaken van SQL DML statements kan de ontwikkelaar gebruik maken van 2 wizards: de SQL Wizard of de SQL Statement Builder voor de meer gevorderde ontwikkelaar. Beide wizards laten toe om via een aantal schermen een volledig SQL DML statement stap-voor-stap op te bouwen, de syntax ervan te controleren en het statement ook uit te voeren. Het resultaat is een bestand met de extensie .sqx. Deze SQL statements kunnen niet enkel uitgevoerd en geëditeerd worden, maar kunnen bovendien gebruikt worden voor een reeks bijzondere acties:

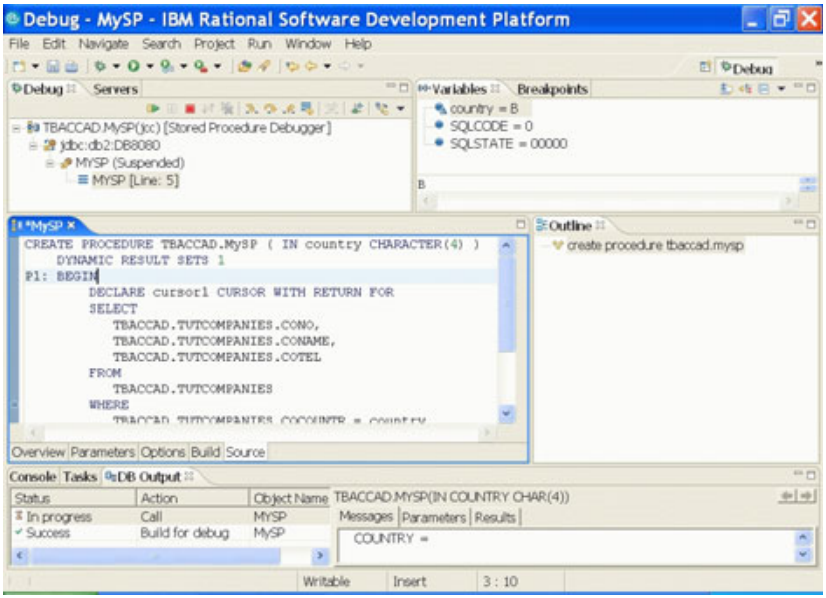
- genereren van een XML-bestand dat gebruikt kan worden met de XMLtoSQL class (insert, update of delete van rijen in een databasetabel gebruik makende van een XML-document)

- genereren van een DADX-bestand dat gebruikt kan worden door de DB2 XML Extender (samenstellen of ontrafelen van data). Deze bestanden worden onder andere gebruikt door web-services
- genereren van een Java bean (uitvoering van het statement)
- gebruik in een relational-to-XML mapping (mappen van een result set van een SQL statement met een XML-bestand). Deze mapping kan op zijn beurt gebruikt worden voor het genereren van een DAD bestand.

RAD en DB2 Stored Procedures

De RAD workbench voorziet in een groot aantal tools, die de ontwikkelaar kunnen helpen bij de ontwikkeling van DB2 SPs (Stored Procedures). De New Stored Procedure wizard laat toe om een Java of SQL SP volledig van nul op te bouwen, of om te vertrekken vanaf een bestaand SQL DML statement. Tijdens de creatie van een SP kunnen ondermeer de naam van de SP, het bijbehorende SQL statement en de parameters aangegeven worden. Achteraf bestaat de mogelijkheid om de SP rechtstreeks op de database server te deployen (build) en ook uit te voeren. Een SQL SP gedefinieerd voor DB2 UDB for LUW kan ook gedeployed worden voor debug-mode. Wanneer je een SP wil debuggen, doe je dit vanuit het Debug perspective. In Figuur 3 zie je hiervan een voorbeeld.

Figuur 3: debugging van een DB2 Stored Procedure.



Het Debug perspective is samengesteld uit meerdere gesynchroniseerde views. In het Debug view kan je stap-voor-stap de SP doorlopen, terwijl in het Variables view de inhoud van de variabelen wordt getoond. Hier kunnen ook wijzigingen aan de inhoud van variabelen aangebracht worden. Tijdens het debuggen worden SQLCODE en SQLSTATE continue gemonitored. Door het aanbrengen van breakpoints (op code of op variabelen) kan de uitvoering van de SP gecontroleerd worden. Deze zijn in de SP editor (centraal in de figuur) zichtbaar naast de code. Het DB Output view toont naast het verloop van de SP ook eventuele input en output van de SP, alsook het resultaat na uitvoering. Debugging is mogelijk voor zowel lokale als remote DB2 servers.

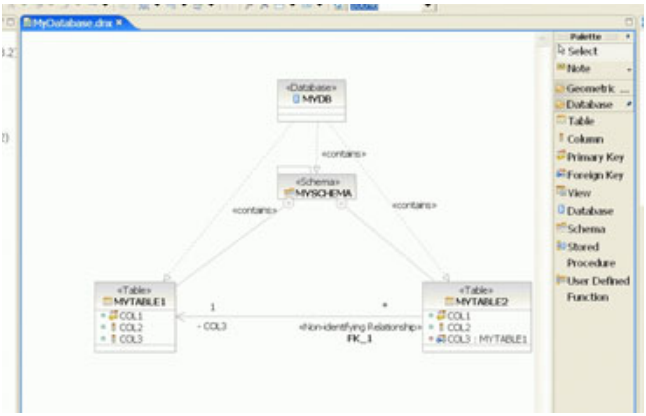
RAD en User Defined Functions

Voor de definitie van SQL UDFs (User Defined Functions) heeft de ontwikkelaar in RAD de keuze tussen een SQL scalar of table UDF (enkel DB2 UDB for LUW). Het aanmaken van een UDF gebeurt via de New DB2 UDF wizard. Net zoals voor DB2 SPs moet de UDF achteraf nog gedeployed worden op de database server alvorens hij uitgevoerd kan worden. Naast de traditionele UDFs kan men in RAD ook enkele bijzondere UDFs aanmaken. De New WebSphere MQ User-defined Function wizard laat toe om een DB2 UDF aan te maken, die in staat is boodschappen van een WebSphere MQ message queue te ontvangen. Het gebruik van zo'n UDF laat toe om de inhoud van een MQ-boodschap over te brengen naar een DB2-tabel door middel van een eenvoudig SQL statement. Een tweede bijzonder type van UDF die je kan aanmaken in RAD, is een UDF die web-services kan benaderen. Via de gepaste wizard specificeer je het WSDL bestand (XML-based formaat voor de beschrijving van de externe interface van de web-service), selecteer je de database waarin je de UDF wenst aan te maken, en specificeer je eventuele opties. Voor elke operatie die gedefinieerd is in het WSDL bestand, wordt 1 UDF aangemaakt.

RAD en visualisatie

Voor het aanmaken van database objecten in RAD kan je gebruik maken van de wizards, zoals hierboven reeds uitvoerig besproken werd. Een andere mogelijkheid is, om te vertrekken van een visualisatie model. Zowel standaard UML, Information Engineering (IE) of Integrated Definition for Information Modeling (IDEF1X) diagrammen kunnen dankzij een aantal grafische editors op een eenvoudige manier samengesteld worden. Het genereren van een database object is dan slechts één stap verder. In Figuur 4 zie je een voorbeeld van een UML-diagram voor de definitie van een nieuwe database, met bijbehorend schema en tabellen in het Data Definition view.

Figuur 4: visualisatie van database-objecten in het Data Definition view



Besluit

De RAD workbench biedt de database-ontwikkelaar/DBA een handig werkinstrument aan voor het uitvoeren van een aantal database-manipulaties. Voor ingewikkelder database-acties zal men echter nog steeds beroep moeten doen op de traditionele database interfaces.

DB2 en Content Management - 2: het data model

Eric Venmans (ABIS)

Inleiding

Een eerste artikel over de DB2 ContMgr (voluit DB2 Content manager) gaf een algemene beschrijving van dit systeem. Nu gaan we dieper in op het *data model* dat ondersteund wordt door de DB2 ContMgr.

Een Data Base Management Systeem (DBMS) kan een *logisch data model* geheel of gedeeltelijk implementeren. Welke elementen uit een *data model* kunnen geïmplementeerd worden, hangt af van de mogelijkheden van het gebruikte DBMS. Enerzijds is het de bedoeling in dit artikel de mogelijkheden en beperkingen van DB2 zelf te bekijken. Anderzijds willen we vooral de extra mogelijkheden belichten die de DB2 ContMgr hieraan toevoegt.

Algemene omschrijving

In een DB2 ContMgr beheert de *Library Server* component het *data model*. Het model kan gezien worden als de verzameling opslagstructuren in een centrale 'repository', te vergelijken met tabeldefinities in een database. De DB2 ContMgr repository bevat minstens de structuur van het geïmplementeerde *data model*. De inhoud, de gegevens zelf, kunnen eveneens in deze repository bewaard worden. 'Kunnen', want dikwijls gaat men de informatie geheel of gedeeltelijk elders bewaren. Dit gebeurt in de eerste plaats door of via de *Resource Managers*. We kunnen de repository meer zien als een gestructureerde verzameling 'metadata' (informatie over informatie). Een mengeling van gegevens en referenties naar gegevens is de meest courante vorm.

Het eenvoudige 'data model'

Een traditioneel *data model* bestaat uit een aantal componenten. Kort samengevat kan men zeggen dat een *data model* bestaat uit een verzameling gerelateerde 'dingen' met bijhorende 'kenmerken'.

Een 'ding' in een data model is een abstract 'iets', een algemene beschrijving van iets waarvan men concrete occurrences (instances) wil gebruiken. De concrete occurrences vinden we terug in bijvoorbeeld een DBMS. Het 'ding' waarvan sprake, kennen we onder zijn technisch naam als o.a. 'entiteit' of als 'object' of beter nog als 'klasse' (een object immers is in een object-georiënteerde benadering reeds een concrete instance van een 'klasse').

In een DBMS wordt een entiteit meestal vertaald of geïmplementeerd door een tabeldefinitie. Voor sommige entiteiten definieert men meerdere tabellen (o.a. omwille van normalisatieregels), maar soms ook gaat men meerdere entiteiten samenvoegen in één enkele tabel (denormalisatie).

Als men over bijhorende kenmerken praat, heeft men het over 'attributen', 'rubrieken', 'properties', ... In een DBMS vinden we ze normaal terug als kolomdefinities van een tabel. Een speciaal kenmerk krijgt extra aandacht:, namelijk 'een verband met, een relatie met'. Hiermee worden tabellen aan elkaar gekoppeld. Indien de verbanden expliciet gedefinieerd worden, controleert en bewaakt het DBMS de manipulaties ervan.

Een *data model* kan ook de beschrijving van 'activiteiten' bevatten, regels i.v.m. insert-, update- en delete-operaties. Hiervoor kan men in een DBMS o.a. 'stored procedures' en 'triggers' definiëren en gebruiken.

Het DB2 data model

DB2 is een erg traditioneel DBMS (we praten hier in de eerste plaats over de z/OS-versie). Het kan zonder problemen een eenvoudig *data model* implementeren, maar ook niet meer. Relatief eenvoudige zaken, die o.a. NIET ondersteund worden zijn:

- Samengestelde attributen.
  Ofwel registreert men een 'adres' als geheel, ofwel registreert men de onderdelen ervan: 'straat', 'postcode', 'gemeente', ...; aangeven dat een 'adres' is samengesteld uit de individuele componenten is (nog) niet ondersteund.
- referenties, links.
  Er is geen ondersteuning voor relaties die anders dan via 'foreign key' definities worden vastgelegd; dit betekent concreet dat men opgesloten blijft in het eigen systeem (geen gecontroleerde verwijzingen naar externe gegevens) en dat de relaties in een aantal gevallen 'te streng' of 'overdreven' gecontroleerd worden.
- Sub- en supertypes.
  Om via overerving entiteiten (of tabellen) te definiëren die afgeleid zijn van anderen (vb. 'arbeider' en 'bediende' als subtypes van het supertype 'werknemer') is er geen rechtstreekse ondersteuning.

Dit betekent concreet dat men aan een DBMS zoals DB2 geprogrammeerde code moet toevoegen om ondersteuning te krijgen voor bovengenoemde faciliteiten. Deze toevoegingen kan men ofwel zelf programmeren ofwel toevoegen via een extra softwarelaag. DB2 ContMgr is zulk een softwarelaag.

Het DB2 ContMgr data model

Dit *data model* is gebouwd bovenop een relationeel DBMS (DB2 of Oracle). De DB2 ContMgr voegt een aantal 'extra' mogelijkheden toe aan het ondersteunende systeem en is in de eerste plaats bedoeld als repository voor het content management.

ITEMS

Het geïmplementeerde *data model* is opgebouwd uit 'items'. Een item is een eenheid van informatie. Het kan vergeleken worden met een 'entiteit', een 'object', m.a.w. met het 'ding' uit het begin van ons artikel.

Een item is een template voor item occurrences, zoals een tabel een template is voor rijen. De template is de eigenlijke *data model* component. Alle item-templates samen beschrijven de structuur van de beheerde content. De item occurrences bevatten de concrete gegevens of referenties ernaar.

Een item in zijn eenvoudigste vorm (zie voorbeeld 1) is te vergelijken met een entiteit waarvoor één tabel gebruikt wordt: een verzameling 'single-value' attributen.

Voorbeeld 1: eenvoudig ITEM

```
Item: BOEK

Attributen:
- identificatie
- ISBN-nummer
- titel
- auteur
- prijs
- voorraad
```

Echter, voor heel wat entiteiten en/of objecten is dit eenvoudig model ontoereikend. Wat als we bijvoorbeeld een BOEK meerdere auteurs heeft. Als we DB2 rechtstreeks gebruiken zijn er een drietal mogelijkheden om dit op te lossen:

- we kunnen per auteur een nieuw attribuut (kolom) definiëren
- we kunnen de namen opnemen in dezelfde kolom (m.a.w. concateneren) met telkens één of andere scheidingsteken (delimiter)
- we kunnen een extra tabel maken (met BOEK.identificatie, auteur) waarin, indien nodig, meerdere rijen per boek worden geregistreerd.

Oplossing 1 en 2 hebben twee in het oogspringende nadelen. Het aantal auteurs dat kan geregistreerd worden, is gelimiteerd (aantal kolommen, capaciteit van de kolom) en vooral het specificeren van zoekargumenten wordt een stuk complexer (meerdere kolommen, substrings van een kolom). De 3de oplossing is vanuit theoretisch standpunt de enige juiste, maar vraagt ook extra inspanningen bij het ophalen van de informatie, zowel van het systeem (verzamelen van informatie uit meerdere structuren), als voor de gebruiker ('join' statements).

De DB2 ContMgr zal ook deze laatste oplossing gebruiken, maar transparant voor de gebruiker. Een item wordt hiervoor opgesplitst

in componenten, die elk een variabel aantal occurrences hebben (zie voorbeeld 2).

COMPONENTEN

Elke component krijgt een eigen onderliggende tabel.

Er is geen theoretische beperking op de complexiteit van dit systeem met componenten:

- een root component kan een onbeperkt aantal child componenten hebben;
- het aantal occurrences van een child component is zo goed als onbeperkt;
- een child component kan op zijn beurt child componenten hebben (nested componenten).

Voorbeeld 2: ITEM met COMPONENTEN

```
Item: BOEK

ROOT component attributen:
- identificatie
- ISBN-nummer
- titel
- prijs
- voorraad

CHILD component attribuut
(meerdere occurrences):
- auteur
```

Er is wel een praktische limiet als performance komt meespelen, hetgeen vroeg of laat zal gebeuren.

Als we een component meer in detail bekijken, kunnen we i.v.m. attributen praten over:

- Systeem-attributen (zij worden geplaatst in de prefix van een component).
  Ze identificeren de component op systeemniveau, ze dienen, indien van toepassing, om te verwijzen naar de parent en root-componenten en ze bevatten systeeminformatie over de attributen van de betrokken component (naam, data type, ...).
- Gebruikers-attributen.
  Dit zijn de traditionele attributen zoals die ook voorkomen in het *logisch data model*; men kan ze groeperen (zie als voorbeeld het adres en zijn bestanddelen) en ze als groep manipuleren.

Voor al deze informatie worden tabellen aangemaakt met extra kolommen voor de systeemattributen. Er is per item één parent-tabel (met alle root component informatie) en één tabel per dependent component. Een tabel voor een dependent component heeft een 'foreign key' definitie. Deze kan verwijzen naar de tabel van de root

component of naar de tabel van een dependent component, hoger in de componentenhiërarchie.

Opmerking hierbij: een item heeft een strikt hiërarchische structuur: men kan enkel één-op- veel relaties definiëren van parent componenten naar hun dependent componenten. Dit wil niet zeggen dat er geen ondersteuning is voor veel-op-veel relaties tussen componenten. Enkel binnen eenzelfde item kan het niet. Veel-op-veel relaties worden op een andere manier geïmplementeerd.

RELATIES tussen COMPONENTEN

Componenten die een verband hebben met elkaar, kunnen hiervoor op verschillende manieren ondersteuning krijgen van het systeem.

De child component 'AUTEUR' uit ons voorbeeld kan men ook als root component definiëren van een apart item. Dit zal men vooral doen als de component meerdere attributen heeft en vooral als de component een verband heeft (of kan hebben) met meerdere occurrences van een andere component ('BOEKEN' in ons voorbeeld). Het verband zelf kan op meerdere manieren vastgelegd worden.

Voorbeeld 3: LINK tussen ITEMS

```
Item: BOEK

ROOT component attributen:
- identificatie
- ISBN-nummer
- titel
- prijs
- voorraad

Virtueel CHILD component:
- AUTEUR
(meerdere occurrences moge-
lijk)

Item: AUTEUR

ROOT component attributen:
- identificatie
- naam
- adres
- geboortedatum
- moedertaal

Virtueel CHILD component: -
- BOEK
(meerdere occurrences moge-
lijk)
```

- De 'link' (zie voorbeeld 3).
  Dit is de traditionele veel-op-veel relatie, waarvoor een extra tabel wordt gedefinieerd. Deze bevat de actuele relaties tussen rijen uit de betrokken tabellen, dus tussen occurrences van de betrokken

componenten. In het DB2 ContMgr systeem is deze tabel zelf niet zichtbaar voor de gebruiker. Hij wordt automatisch gedefinieerd door de DB2 ContMgr in het ondersteunende DBMS. De 'link' kan enkel gedefinieerd worden tussen root-componenten. Occurrences van deze root-componenten kunnen ook onafhankelijk van elkaar voorkomen.

- De 'referentie' (zie voorbeeld 4).
  Deze is te vergelijken met de traditionele 'foreign key' relatie (één-op-veel), maar dit keer tussen items. De 'target' van de referentie is altijd een ROOT-component. De 'source' van de relatie is ofwel een ROOT-component (voor één-op-één relaties) ofwel een CHILD-component (voor één-op-veel relaties). Zulk een 'source' CHILD-component kan naast de referentie ook eigen attributen hebben.
  De 'referentie' wordt bewaakt door het systeem. Afhankelijk van de specificaties hebben we te maken met telkens één van de traditionele 'delete'-regels: restrict, cascade, set null of no action.

Voorbeeld 4: referentie vanuit een CHILD COMPONENT

```
Item: BOEK

ROOT component attributen:
- identificatie
- ISBN-nummer
- titel
- prijs
- voorraad

CHILD component attributen:
- AUTEUR.identificatie
- bijdragestatus

Item: AUTEUR

ROOT component attributen:
- identificatie
- naam
- adres
- geboortedatum
- moedertaal
```

- De 'foreign key''.
  In de context van de DB2 ContMgr wordt het concept 'foreign key' gebruikt om te verwijzen naar gegevens die buiten de *Library Server* worden beheerd. Het is een 'pointer'-attribuut dat de identificatie bevat van een 'extern' stuk informatie.

ITEM OCCURRENCES

Tot nu toe hebben we ons binnen het DB2 ContMgr data model beperkt tot structuren. Nu staan we even stil bij de 'occurrences', de gegevens zelf. Zoals reeds vroeger aangegeven, wordt de inhoud van items ofwel beheerd door de *Library Server* zelf, of door één van de

Resource Managers. Het verschil is merkbaar binnen de DB2 ContMgr door de 'aard' van een item te bekijken.

- Normaal item.
  Bij deze categorie van items wordt alle informatie beheerd door de *Library Server*. De informatie is terug te vinden in de tabellen van het ondersteunende DBMS. Het gaat hier in de eerste plaats over traditionele gestructureerde informatie (waaronder ook LOBs). Deze gestructureerde informatie zit mogelijk al geheel of gedeeltelijk in andere 'traditionele' databases. Eventueel kunnen ze via 'foreign keys' geïntegreerd worden zonder ze te moeten dupliceren.
- Resource item.
  Bij deze categorie spreken we over 'objecten' die door één van de *Resource Managers* beheerd worden. Het 'item' bevat informatie over het object zodat het kan gelokaliseerd en geselecteerd worden. Het kan via links en referenties verbonden zijn met andere items uit het *data model*. Een resource item kan een willekeurig formaat hebben: HTML, plain TEXT, bit map, JPEG, GIF, spreadsheet, ... Het formaat wordt aangegeven in de DB2 ContMgr via het overeenkomstige MIME-type (internet standaard). Op deze manier kan het systeem de juiste software gebruiken voor manipulaties van het betrokken resource item.
- Een document.
  Dit is geen aparte categorie, maar een variante op resource items. Een document zelf is geen resource item. Het verwijst niet rechtstreeks naar een extern object, Een document item wordt volledig beheerd door de *Library Server*, maar bestaat steeds uit een verzameling referenties naar 'document parts'. Elk document part is wel een resource item, verwijzend naar een extern object. Een document geeft m.a.w. toegang tot een verzameling objecten.

Voorlopige afronding

Nu we beschreven hebben hoe men in een DB2 ContMgr informatie 'logisch' organiseert en waar men het bewaart, zijn we klaar om in een laatste artikel het gebruik van 'content' toe te lichten:

- hoe kan men (snel) geregistreerde informatie terugvinden?
- hoe wordt informatie gemanipuleerd en gecontroleerd in het DB2 ContMgr systeem?

CURSUSPLANNING APR - JUN 2006

DB2 for z/OS, een totaaloverzicht	1825 EUR	03-07/04 (W), 29/05-02/06 (W), 19-23/06 (L), 24-28/07 (W)
DB2 UDB, een totaaloverzicht	1750 EUR	29/05-02/06 (W)
RDBMS concepten	350 EUR	03/04 (W), 29/05(W), 19/06 (L), 24/07 (W)
Basiskennis SQL	350 EUR	04/04 (W), 30/05(W), 20/06 (L), 25/07 (W)
DB2 for z/OS basiscursus	1125 EUR	05-07/04 (W), 31/05-02/06 (W), 21-23/06 (L),
DB2 UDB basiscursus	1050 EUR	31/05-02/06 (W)
SQL workshop	750 EUR	18-19/04 (L), 02-03/05 (W)
Extended SQL in DB2	425 EUR	20/04 (L)
DB2 for z/OS programmering voor gevorderden	800 EUR	10-11/04 (L)
DB2 for OS/390: SQL performance	1275 EUR	22-24/05 (W)
XML in DB2	425 EUR	21/06 (L)
DB2 for z/OS database administratie	1700 EUR	12-15/06 (L)
DB2 for z/OS operations and recovery	1425 EUR	14-16/06 (L)
DB2 for z/OS DBA and operations	2175 EUR	12-16/06 (L)
DB2 for z/OS in een Java omgeving	425 EUR	04/04 (W), 27/06 (L)

Plaats: L = Leuven; W = Woerden; details en extra cursussen: www.abis.be

Postbus 220
Diestsevest 32
BE-3000 Leuven
Tel. 016/245610
Fax 016/245691
training@abis.be

abis
TRAINING & CONSULTING

Postbus 122
Pelmolenlaan 1-K
NL-3440 AC Woerden
Tel. 0348-435570
Fax 0348-432493
training@abis.be