ABIS NIEUWSBRIEF VOOR DB2 PROFESSIONALS

JAARGANG 11 - NUMMER 1, JUNI 2018

OPEN CURSOR

IDUG Lissabon is alweer een hele poos achter de rug, en we kijken alvast uit naar IDUG Malta. Zoals verwacht leerden we in Lissabon alles over Db2 12 for z/OS en over Db2 11 for LUW.
Bovendien was er een afzonderlijke "analytics" track: data science (of AI, of BI, of Big Data, of hoe het vandaag ook genoemd wordt) blijft dus prominent aanwezig voor al wie met data bezig is.
Geïnspireerd door o.a. deze lezingen, hebben we voor u in dit nummer van Exploring Db2 opnieuw enkele onderwerpen uitgediept: nieuwe mogelijkheden van Db2, voor de DBA, de applicatie-analist, de eindgebruiker. Telkens met een kritische blik onder de loep genomen. En hapklaar gemaakt voor u!
Vergeet ook niet, ons cursusaanbod te bekijken op de laatste pagina.

Alvast veel leesgenot,
Het ABIS Db2-team.

IN DIT NUMMER:

- Een update over het gebruik van IDz (alias RDz), in "*IDz voor ontwikkelaar & DBA*".
- Een vervolg op de vorige bijdrage over JSON en XML: "*JSON - in Db2*", met deze keer de Db2-specifieke aspecten, zowel server als client.
- Een derde bijdrage heeft het over Db2's nieuwe model van versionering: "*Db2 12 Continuous Delivery - of toch #13?*"
- En ten slotte, in "Dossier 12", gaat het deze keer over de nieuwe zogenaamde "advanced triggers".

1

CLOSE CURSOR

Ook in de volgende nummers van Exploring Db2 blijven we u op de hoogte houden van alle belangrijke ontwikkelingen i.v.m. Db2, en ook (in de ruimere zin) van data(bases) en data analytics.

Over welke onderwerpen wilt u meer lezen? Meld het ons: training@abis.be

IDz voor ontwikkelaar & DBA

Gie Indesteege (ABIS)

In *een vorig artikel* hebben we de basismogelijkheden beschreven om Db2-applicaties voor z/OS te ontwikkelen met behulp van RDz: *Rational Developer for System z*. Een grafische ontwikkelomgeving, gebaseerd op Eclipse, om zowel COBOL/PLI programma's te bouwen, als om databasetoegang voor te bereiden.

Ondertussen is het tool van naam veranderd: *IBM developer for z System* (**IDz**), en bovendien geïntegreerd met de Application Delivery Foundation (**ADF**). Dat betekent concreet dat naast ontwikkeling van toepassingen, ook data-analyse, data discovery, file manager, fout-analyse, performance-analyse, ... op een eenvoudige manier kunnen gerealiseerd worden.

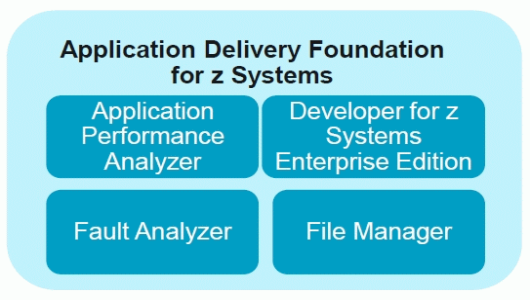
In dit artikel willen we even stil staan bij de belangrijkste (extra) mogelijkheden, met betrekking tot data access, zowel vanuit het standpunt van de ontwikkelaar als van de DBA. Het bouwen van applicaties met embedded SQL werd al behandeld in het vorige artikel.

Wat is IDz?

IBM developer for z Systems (Enterprise Edition) biedt enerzijds de nodige ontwikkelomgeving voor traditionele COBOL/PLI applicaties (batch en online), anderzijds ook interactieve test- en debug-mogelijkheden.

Toegang tot data is mogelijk voor zowel de klassieke z/OS filesystemen (sequentieel, partitioned, VSAM) als voor de databases IMS en Db2. Voorbereiden van (test)data kan gerealiseerd worden met behulp van de File Manager integratie.

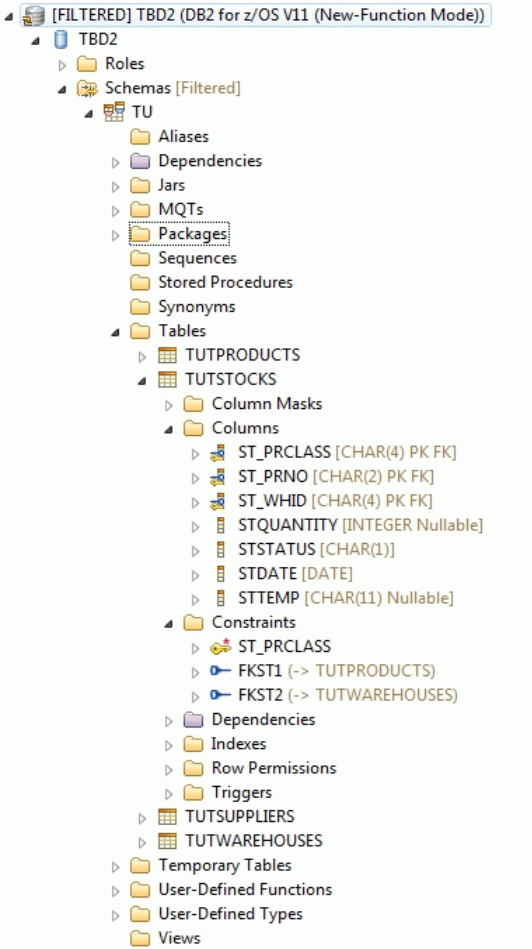
Programma-analyse en (statische) data flow analyse zijn nu standaard voorzien in IDz.



Data-analyse

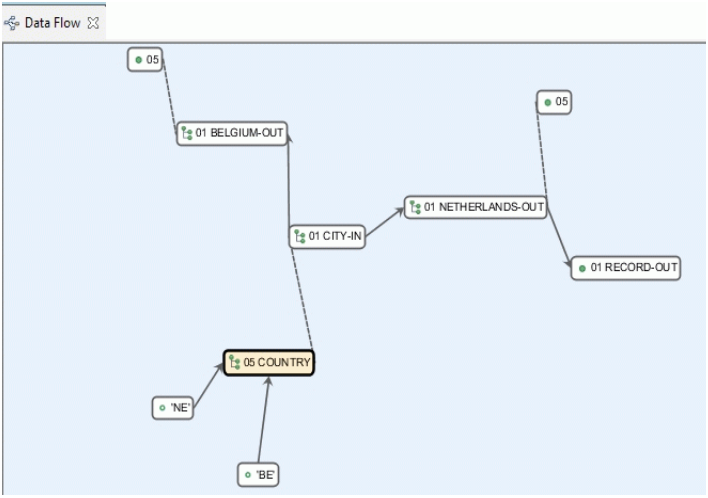
Bij de analyse en design van een applicatie vertrekt men vaak vanuit het datamodel. Hoe zijn de data georganiseerd, welke data worden in de toepassing gebruikt, welke data worden aangeleverd, welke data komen in het resultaat? Dat betekent dat datastructuur en data flow in grote mate de kwaliteit van een goede applicatie bepalen.

De IDz-ontwikkelaar kan via het Data Perspectief en de **Data Source Explorer** View een gedetailleerd beeld krijgen van de database-structuren, om van daaruit de nodige SQL-queries op te bouwen.



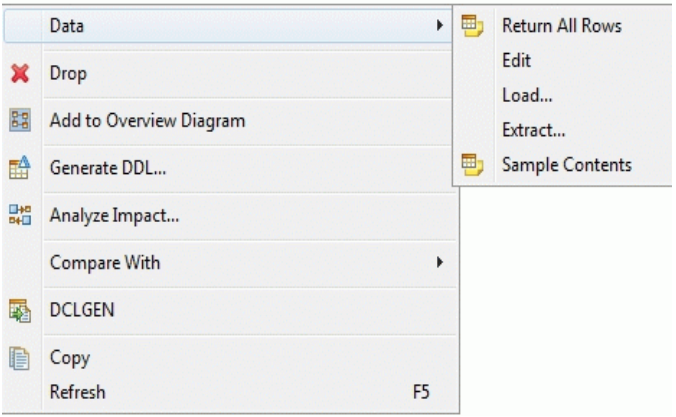
Voor elke tabel is dus voor elk van z'n kolommen (inclusief eventuele hidden columns) de kolomdefinitie, de constraints en de inhoud beschikbaar.

Vanuit het (COBOL/PLI) programma zelf kan de ontwikkelaar de **data flow** volgen.



Voor elk 'element' kan een link gevraagd worden naar de instructie die het 'element' benadert. Het visualiseren helpt om een impact- of data flow-analyse te maken.

De DBA op zijn beurt kan gebruik maken van IDz om de database-structuren te definiëren en/of aan te passen. Visualisatie, generatie/definitie van DDL, of impactanalyse is mogelijk, weer vanuit de Data Source Explorer view.



Testen en debuggen

Zoals reeds vermeld is het uittesten van queries, en het debuggen van volledige programma's, inclusief stored procedures, in IDz mogelijk, dankzij de integratie met het Debug Tool voor z/OS (of IBM Debug for z).

De huidige versie van IDz voorziet nog een aantal extra features, zoals

- visuele debugging (via de Program Control Flow view);
- gebruik van zUnit framework, ook voor Db2 batch programma's;
- GUI of 3270 debugging;
- speciale breakpoint-definities;
- code coverage.

Tot slot

IDz gebruiken, in combinatie met de andere producten uit de Application Delivery Foundation, biedt zowel de ontwikkelaar als de DBA de nodige hulpmiddelen om efficiënt databases en applicaties te organiseren, te ontwikkelen, te testen en te deployen. Dit laatste aspect zal in een toekomstig artikel verder worden toegelicht.

Voor meer details verwijzen we naar de IBM product-documentatie:

https://developer.ibm.com/mainframe/products/ibm-developer-for-z-systems-enterprise-edition/

JSON - en Db2 (II)

Peter Vanroose (ABIS)

In *het vorige nummer* van Exploring Db2 kon u lezen wat het JSON-dataformaat precies is en wat de belangrijkste verschillen zijn met XML.

In deze bijdrage wordt ingegaan op de “native” ondersteuning voor het JSON-formaat door Db2.

Overzicht

Net zoals voor de XML-ondersteuning die Db2 sinds versie 9 biedt, geldt de JSON-ondersteuning zowel voor Db2 LUW als voor Db2 z/OS. JSON-support is beschikbaar sinds Db2 11 voor z/OS (of Db2 10 met APAR II14727) en sinds Db2 10.5 FP3 voor LUW.

In tegenstelling tot de situatie voor XML, dat een (toen nieuw) datatype XML gebruikt, worden JSON-documenten in Db2 gewoon als tekst voorgesteld, dus via datatype varchar of CLOB. Alle JSON-gerelateerde ondersteuning door Db2 gebeurt enerzijds d.m.v. een aantal SQL-functies (server-side dus), i.h.b. `JSON_VAL`, `JSON_TABLE`, `JSON2BSON` en `BSON2JSON`. Deze functies bevinden zich in schema `SYSTOOLS`. Anderzijds biedt Db2 vooral client-side support voor JSON, onder de vorm van Java-classes (in `db2nosql.jar`), en van een Db2 for LUW command-line interface genaamd `db2nosql`.

Ter herinnering: op het “laagste” niveau bevat een JSON-document zgn. name/value paren; op een hoger niveau kunnen values op hun beurt bestaan uit arrays van name/value paren. Name en value worden gescheiden door een “`:`”; een lijst van name/value paren wordt omsloten door “`{}`”, en arrays van dergelijke lijsten worden omsloten door “`[]`”.

BSON

Om efficiëntie-redenen gebruikt Db2 intern het zgn. *BSON-formaat*: binair JSON (of eigenlijk de IBM-variant: extended binary JSON). Alle JSON-functies, ook de niet hierboven genoemde, werken met dit BSON-formaat; enkel de twee functies `JSON2BSON` en `BSON2JSON` converteren tussen BSON en het leesbare JSON. Een Db2-kolom voor JSON-opslag moet dus van datatype `BLOB` zijn; elke INSERT of UPDATE in een dergelijke JSON-kolom moet dan via de functie `JSON2BSON` gaan, en elke SELECT heeft een expliciete `BSON2JSON` nodig. Voorbeeld:

```
CREATE TABLE courseinfo ( data BLOB(10K) );
INSERT INTO courseinfo (data)       VALUES (SYSTOOLS.JSON2BSON(
         '{CourseParticipants: { Participant: [
            {firstName: "Maria", lastName: "Meuris"},
            {firstName: "Bert",  lastName: "Mak"   },
            {firstName: "Peter", lastName: "Buenk" } ] } }'
    ));
SELECT SYSTOOLS.BSON2JSON(data) FROM courseinfo;
```

JSON_VAL

Echte Db2-ondersteuning voor JSON begint pas wanneer we meer kunnen vragen dan gewoon een volledig JSON-document opslaan in een Db2-tabel en daarna terughalen uit die tabel.

Hier komt in eerste instantie de functie `JSON_VAL` van pas: daarmee kan uit een BSON-document, op basis van een bepaalde "name", de bijhorende "value" opgehaald worden. (In Db2 voor LUW heet deze functie trouwens `JSON_VAL2`.)

Een voorbeeld verduidelijkt dit:

In het voorbeeld hierboven is "`CourseParticipants`" de enige "name" op het hoogste niveau, maar z'n "value" is op z'n beurt een name/value paar met name "`Participant`", waarvan de value een array van name/value paren is.

De waarde "`Meuris`" is dan een atomaire value die hoort bij de name `CourseParticipants.Participant.lastName`

Met behulp van de functie `JSON_VAL` kunnen we "`Meuris`" dus extraheren als volgt:

```
SELECT SYSTOOLS.JSON_VAL
       ( data
       , 'CourseParticipants.Participant.lastName'
       , 's:64'                                      ) AS lastName
FROM courseinfo;
```

Bemerk dat `JSON_VAL` drie argumenten nodig heeft: eerst uiteraard de BSON-data (het document) zelf; het tweede argument vermeldt de gewenste "name". Het derde argument geeft aan naar welk SQL-datatype de waarde moet geconverteerd worden.

"`s:64`" (of meer algemeen: "`s:n`" voor een bepaald geheel getal `n`) staat voor type `VARCHAR(n)`; andere specificaties zijn "`i`" voor `INTEGER`, "`ts`" voor `TIMESTAMP`, "`d`" voor `DATE` en "`t`" voor `TIME`. (Andere datatypes zijn o.a.: "`l`" voor `BIGINT`, "`n`" voor `DECFLOAT`, "`f`" voor `DOUBLE`, en "`b`" voor `VARBINARY`.)

In bovenstaand voorbeeld hebben we een klein bijkomend "probleempje", indien we enkel de naam "Meuris" willen: ook de andere twee lastNames "`Mak`" en "`Buenk`" hebben dezelfde "`CourseParticipants.Participant.lastName`" als "name".

Anders gezegd: we verwachten(*) dat

```
JSON_VAL(data, 'CourseParticipants.Participant.lastName', 's:64')
```

een array van lengte 3 teruggeeft; de eerste waarde van deze array kunnen we als volgt bekomen: (bemerk de extra ".0")

```
SELECT SYSTOOLS.JSON_VAL
       ( data
       , 'CourseParticipants.Participant.0.lastName'
       , 's:64'  )
FROM courseinfo;
```

Inderdaad: de "value" die bij `CourseParticipants.Participant` hoort in het voorbeeld-document is zelf een array, en daarvan willen we de eerste entry, waarvan we dan de name "`lastName`" willen opvragen. JSON-indexering van een array begint dus blijkbaar bij 0, niet bij 1.

Anderzijds kunnen we uiteraard ook niet-atomaire data uit een JSON-document extraheren m.b.v. `JSON_VAL`. In dat geval zal een JSON-document teruggegeven worden met de gevraagde data. Als we b.v. de value vragen die bij name "`CourseParticipants.Participant`" hoort, dus:

```
SELECT SYSTOOLS.JSON_VAL
       (data, 'CourseParticipants.Participant', 's:1024')
FROM courseinfo;
```

dan krijgen we de volgende JSON-array (als `VARCHAR(1024)`) terug:

```
[  {"firstName": "Maria", "lastName": "Meuris"},
   {"firstName": "Bert",  "lastName": "Mak"   },
   {"firstName": "Peter", "lastName": "Buenk" }
]
```

JSON_TABLE

Het zou handiger zijn om een volledige lijst van values terug te krijgen in tabelvorm. Daarvoor is de functie `JSON_TABLE` bedoeld: deze tabelfunctie geeft een tabel terug van twee kolommen met kolomnamen `TYPE` en `VALUE`. Die tweede kolom bevat dan de lijst van values:

```
SELECT j.type, j.value
FROM   courseinfo c,
       TABLE(SYSTOOLS.JSON_TABLE(c.data,
              'CourseParticipants.Participant.lastName', 's:1024')) j;
```

resultaat:

```
TYPE        VALUE
   2        Meuris
   2        Mak
   2        Buenk
```

De kolom `VALUE` is technisch gesproken altijd van datatype `VARCHAR`; maar afhankelijk van het derde argument (hier: `s:1024`) zal de waarde van de kolom `TYPE` (integer) verschillend zijn: 2 staat dus voor "string" data; type 16 betekent "integer", en 17 komt overeen met timestamp. Een type 3 betekent dat de waarde in kolom `VALUE` nog een JSON-(sub)document is. Dit zou b.v. het geval zijn voor de output van

```
TABLE(SYSTOOLS.JSON_TABLE(data, 'CourseParticipants.Participant', 's:1024'))
```

met name:

```
TYPE        VALUE
   3        {firstName:"Maria",lastName:"Meuris"}
   3        {firstName:"Bert",lastName:"Mak"}
   3        {firstName:"Peter",lastName:"Buenk"}
```

db2nosql

De command-line interface `db2nosql.sh` (op Linux/Unix) of `db2nosql.bat` (op Windows) zorgt voor een "native JSON" front-end user interface naar een Db2 database met BSON-kolommen. De nodige server-side conversies en functie-oproepen van `JSON_VAL` en `JSON_TABLE` worden door deze client-side interface verborgen. Hiervoor moet (eenmalig) de database "enabled" worden voor gebruik vanuit `db2nosql` m.b.v. de instructie "`enable(true)`".

(Het enige wat deze "enablement" doet, is het creëren van de genoemde JSON-functies in schema `SYSTOOLS`.)

```
C:\temp> db2nosql.bat -db sample
JSON Command Shell Setup and Launcher.
IBM DB2 NoSQL JSON API 1.1.0.0 build 1.4.245
Type your JSON query and hit <ENTER>
nosql> enable(true)
Executing SQL...
CDJSN1209I Database artifacts created successfully.
nosql> db.courseinfo.insert(
        { CourseParticipants: { Participant: [
          { firstName: "Maria", lastName: "Meuris" },
          { firstName: "Bert",  lastName: "Mak"    },
          { firstName: "Peter", lastName: "Buenk"  }
        ] } }
       )
CDJSN1000I Command successfully completed.
nosql> quit
```

Dit nosql-interface-command `db.*.insert` zorgt voor de nodige Db2-objecten "achter de schermen"; zo moeten we enkel een zgn. collectie specificeren; in dit voorbeeld is dat "`courseinfo`". Achter de schermen wordt dat uiteraard een tabel met als naam `TEST."courseinfo"` (met quotes want hoofdlettergevoelig!) Deze tabel heeft een BSON-kolom (altijd genaamd `DATA`).

Het ondervragen van die tabel, via de nosql-front-end, kan dan gebeuren via de instructie "`$find`", waarvoor de overeenkomstige SELECT gegenereerd wordt:

```
nosql> db.courseinfo.$find({})
```

geeft de hele tabel terug; om enkel de rijen te krijgen waarvan één van de "`lastName`"-velden gelijk is aan "`Meuris`":

```
nosql> db.courseinfo.$find({"CourseParticipants.Participant.lastName":"Meuris"})
```

Om enkel de "`lastName`"-velden te krijgen van alle rijen:

```
nosql> db.courseinfo.$find({}, {"CourseParticipants.Participant.lastName": 1})
```

Het eerste argument van `$find` is dus essentieel een horizontale filtering (`WHERE`-conditie), terwijl het optionele tweede argument voor verticale filtering zorgt.

De nosql-interface heeft verder uiteraard nog bijkomende filter- en aggregatie-mogelijkheden (zoals b.v. "`$lt`" voor numeriek "kleiner dan", of "`$group`" voor groeperings-specificaties).

Zie *http://ibm.co/GYfi3e* voor meer details.

Conclusie

Uiteraard gaat de JSON-ondersteuning door Db2 een stuk verder dan wat hier beschreven werd. Zo biedt Db2 o.a. index-ondersteuning (index on expression) voor BSON-kolommen. Verder hebben we het niet gehad over de (eigenlijk belangrijkste) user interface, nl. de Java API. Zie hiervoor *http://ibm.co/19RWv5Y*

Deze bijdrage wou u vooral inzicht geven in de (meestal onzichtbare) ingrediënten achter de schermen, die in de zgn. "nosql" Java-interface niet zichtbaar zijn: de BSON-kolom (meestal met de naam `DATA`) en de JSON-functies in schema `SYSTOOLS`, in het bijzonder `JSON_TABLE` en `JSON_VAL` (of `JSON_VAL2` op LUW).

(*) Voetnoot:

`JSON_VAL` geeft nooit een array terug: het derde argument geeft aan naar welk datatype het tussentijdse resultaat moet geconverteerd worden. Dat datatype is altijd atomair. Wanneer het tussenresultaat een JSON-structuur is, zal die conversie falen en geeft `JSON_VAL` de waarde NULL terug. Een NULL wordt eveneens teruggegeven wanneer de conversie niet mogelijk is, b.v. wanneer als datatype "`i`" wordt opgegeven terwijl de atomaire waarde geen geldige integer is. Wanneer het tussenresultaat een *array* is van atomaire data, dan wordt het *eerste element* van die array teruggegeven. Voor het voorbeeld hierboven zullen dus zowel

```
JSON_VAL(data,'CourseParticipants.Participant.lastName','s:64')
```

als

```
JSON_VAL(data,'CourseParticipants.Participant.0.lastName','s:64')
```

de waarde "`Meuris`" teruggeven. De laatste vorm van de vraag is echter preciezer, en dus te verkiezen. Vervang 0 door 1 of 2 om één van de andere array-elementen (en hun `lastName`) terug te krijgen.

Db2 12 Continuous Delivery - of toch #13?

Kris Van Thillo (ABIS)

Stel je even voor: je bent een sucessvolle commerciële software-leverancier; en om de zoveel maanden breng je een nieuwe versie of release op de markt van één van de belangrijkste relationele databasesystemen op dit moment in gebruik. Dan komt ooit - of je nu Oracle Corporation bent of IBM - het onmogelijke dillema ... breng je een release 13 op de markt? Sommigen onder ons zullen zich ongetwijfeld het probleem van de 13 rode bollen op de staart van de SN Brussels Airlines vliegtuigen herinneren, toch?

Maar wat doe je eraan - na 12 komt meestal 13 - zelfs voor de creatievelingen onder ons ... Een nummer overslaan? Een marketing 'label' met je release associëren om de aandacht af te wenden van? Of gewoon het risico nemen - *all ahead full*?

Agile software ontwikkeling & continuous delivery

Software-ontwikkeling op basis van Agile-methodologieën is ondertussen quasi gemeengoed. We verwachten allemaal dat nieuwe features, correcties aan software 'onmiddellijk' en op een volledig 'transparante' wijze beschikbaar worden. Dit is zo voor gebruikerssoftware, applicatie-software - en we zijn nu reeds zover dat we dit blijkbaar ook verwachten van bedrijfskritische 'systeem' software!

Althans, daar gaan we met z'n allen van uit ...

Db2 12 Continuous Delivery

Het idee is eigenlijk relatief eenvoudig.

In het verleden was het vaak zo dat echt nieuwe Db2 - DB2? - features werden geïntroduceerd bij het inplementeren van nieuwe releases. Basisidee was dat wijzigingen die daarna aangebracht werden op een release eigenlijk steeds 'service' correcties waren op eigenschappen van de release zelf. Waarbij het woord correctie relatief 'breed' kon worden geïnterpreteerd: een correctie omdat de functionaliteit niet deed wat formeel werd beschreven; of omdat door het Db2-lab 'stress' tests niet onmiddellijk werden uitgevoerd in een omgeving die als representatief kan worden beschouwd voor de omgeving waar u als lezer van dit artikel dagelijks mee wordt geconfronteerd.

Dit moeten we natuurlijk ook toegegeven: in de laatste releases van Db2 werden via 'service'-correcties (APARs) ook wel nieuwe features toegevoegd - back-ported - b.v. JSON-support, RESTful services, ...

Maar dus nu: een totaal nieuwe aanpak!

Nieuw?

Beginnend met Db2 versie 12 is Continuous Delivery het nieuwe mantra. In één 'service' ingreep zal veel meer dan vroeger nieuwe functionaliteit worden opgenomen. Of deze nieuwe functionaliteit beschikbaar en/of bruikbaar is, hangt uiteindelijk af van de beheerder van uw Db2-systeem.

Om e.e.a. effectief te kunnen implementeren, werd het concept 'function level', dat reeds in Db2 v8 geïntroduceerd werd, verder uitgebreid. In Db2 12 concreet te interpreteren als een *parameter* die toelaat aan te geven welk van deze 'nieuwe' features te activeren. Op deze manier kan nieuwe functionaliteit via 'service' bekomen worden; geactiveerd indien nodig; genegeerd indien wenselijk.

Verschillende nieuwe features zullen met specifieke function levels worden geassocieerd; afhankelijkheden tussen functions levels zijn mogelijk en waar relevant wordt daar bij activeren van function levels automatisch rekening mee gehouden.

Van zodra specifieke features getest zijn, klaar voor gebruik, kunnen deze als standaard worden opgepikt/gebruikt in applicaties na een REBIND. De bind-parameter `APPLCOMPAT` speelt in deze context een belangrijke rol; concreet, laat ze toe aan te geven welke applicatie welke nieuwe features eigen aan specifieke function levels kan gebruiken.

Maar waarom?

Vraag blijft: waarom deze keuze?

Om klanten de mogelijkheid te geven om aan hun eigen tempo nieuwe functionaliteit te beginnen implementeren. Eigen tempo, vaak dus sneller dan nu; maar dus ook uitstellen. De impact van de migratie naar functionaliteit wordt op deze manier makkelijker controleerbaar en beheersbaar.

Daarnaast is het installereren van een echte nieuwe release vaak een complexe en storende ingreep, met systeemonbeschikbaarheid tot gevolg. Wat bij deze nieuwe benadering eveneens kan worden vermeden.

En om op het niveau van applicaties - de 'bind' dus - nieuwe functionaliteit selectief beschikbaar te kunnen maken; na testen en na evaluatie van betrokken functionaliteit.

Opnieuw - v13?

Of er ooit een Db2 vNext komt? Geen idee, en niet relevant. IBM geeft aan dat de kans klein is dat dit snel zal gebeuren.

Wat relevant is, is dat we ons allen bewust moeten zijn dat software releases, ook als het om z/OS (sub)systeem-specifieke software gaat, op een nieuwe wijze zal worden beschikbaar gesteld. En dat e.e.a. door specialisten op een correcte wijze wordt verwerkt.

.

DOSSIER 12

Over 'basis' triggers en 'advanced' triggers

Db2 ondersteunt reeds lang triggers, maar een onderscheid tussen 'basis' en 'advanced' triggers werd nooit gemaakt. Vanaf Db2 12 is dit echter wel het geval.

Een *basis trigger* is een standaard Db2 11 trigger. Deze ondersteunt een basis-set van SQL-statements, en wordt aangemaakt met de gekende optie *MODE DB2SQL* in het `CREATE TRIGGER` statement.

Een *advanced trigger* is beschikbaar na installatie van Db2 v12 vanaf functie level 500 - te vergelijken met het vroegere 'New function mode'. Concreet wordt het dan mogelijk, SQL PL te gebruiken in de trigger-definitie. Meer specifiek kan men:

- SQL-variabelen in de trigger body opnemen;
- nieuwe SQL-statements - inclusief dynamische SQL - opnemen in de trigger-code;
- SQL PL control statements (IF, WHILE, ...) opnemen;
- gebruik maken van zgn. globale variabelen;
- specifieke bind-opties meegeven bij aanmaak van de trigger (b.v. `EXPLAIN YES`);
- commentaarlijnen toevoegen met `/* ... */`;
- verschillende versies maken van één trigger (waarvan slechts één actief is).

Merk op dat ook voor advanced triggers het `ALTER TRIGGER` DDL statement beschikbaar blijft; alsook de nieuwe syntactische mogelijkheid `CREATE OR REPLACE TRIGGER` (vooral gekend uit een Oracle-omgeving).

Kris Van Thillo

CURSUSPLANNING, JUNI - DEC 2018

SQL & relationele databases: basiskennis	850 EUR	30.08(L), 18.10(W), 08.11(L), 17.12(W)
Db2 for z/OS basiscursus	1425 EUR	24.09(L), 12.11(W)
Db2 for LUW basiscursus	1425 EUR	24.09(L), 12.11(W)
SQL workshop	900 EUR	25.06(W), 24.09(L), 08.11(W), 17.12(L)
SQL voor gevorderden	500 EUR	16.10(L), 06.12(W)
SQL voor BI en Data Science	950 EUR	08.10(L), 13.12(W)
SQL PL database programmeren	1000 EUR	27.09(L)
Db2 triggers, stored procedures, en User-Defined Functions	500 EUR	25.10(W)
Db2 for z/OS: programmeren voor gevorderden	1000 EUR	22.10(L)
Db2 for z/OS: SQL performance	1500 EUR	18.06(L), 21.11(L)
XML in Db2	500 EUR	24.10(L)
Db2 for z/OS: database administratie	2100 EUR	26.11(W), 17.12(L)
Db2 for z/OS: installation & migration	1080 EUR	op aanvraag
Db2 for z/OS: data recovery	1080 EUR	16.08(UK)
Db2 for z/OS: using RACF	540 EUR	op aanvraag
Db2 for z/OS: monitoring & tuning systems' performance	1050 EUR	op aanvraag
Db2 for LUW DBA - Kernvaardigheden	2000 EUR	25.06(W), 13.12(W)
JDBC	500 EUR	05.10(L)
Db2 11 for z/OS: changes & new features	525 EUR	op aanvraag
Db2 12 for z/OS: changes & new features	525 EUR	04.09(W), 29.11(L)
Db2 for LUW V10: new features	500 EUR	op aanvraag
Moderne data warehousing & BI	500 EUR	30.08(W), 19.11(L)
Dimensionaal modelleren	2120 EUR	04.09, 11.12 (Hilversum)
Big Data architectuur & instrastructuur	500 EUR	03.09(W), 04.10(L), 09.11(W), 30.11(L)
Big Data in de praktijk: text analytics	475 EUR	26.10(W), 21.12(L)
Regular expressions	275 EUR	21.09(L), 29.10(W) (avondsess.)

Plaats: (L) = Leuven, (W) = Woerden, (UK) = High Wycombe (bij Londen);
alle cursussen ook beschikbaar op aanvraag en/of op uw lokatie;
Voor details en andere cursussen, zie `http://www.abis.be/html/nlall.html`
Pour détails et d'autres cours, voir `http://www.abis.be/html/frall.html`
For details and other courses, see `http://www.abis.be/html/enall.html`