ABIS NIEUWSBRIEF VOOR DB2 PROFESSIONALS

JAARGANG 4 - NUMMER 5, DECEMBER 2006


## OPEN CURSOR

*Iets later dan gepland hier dan toch nog het laatste nummer van de vierde jaargang.*
*Net terug van IDUG-2006 in Wenen, met in de reiskoffer alweer heel wat informatie over nieuwe ontwikkelingen binnen DB2, en nuttige ideeën en tips voor het verbeteren en optimaliseren van queries.*
*Als belangrijkste blikvanger onthouden we de 'end-of-service'-datum maart 2008 voor DB2 v7 voor z/OS, en het iets vagere 'spring 2007' voor de release van versie 9.*
*DB2 is dus volop in evolutie. Een goede zaak natuurlijk, maar in de eerste plaats rekenen we toch nog altijd op z'n stabiliteit, 7x24 beschikbaarheid en robuustheid. Met hier en daar een 'modernere' look, zoals u in dit nummer kunt ontdekken.*

*Het ABIS DB2-team.*

## IN DIT NUMMER:



## CLOSE CURSOR

In het volgende nummer vindt u meer recursieve SQL, en verder staan we stil bij enkele nieuwe (en oude) mogelijkheden van data-utilities.

Tot dan!

# Gaten in tabellen: recursieve SQL in DB2

*Geert Vandevenne (ABIS)*

Eén van de nieuwe mogelijkheden die DB2 voor z/OS V8 voorziet is recursieve SQL. In DB2 voor Linux, UNIX en Windows bestond dit al sinds versie 7. Recursieve SQL is een krachtige query-techniek die o.a. toelaat om hiërarchisch gestructureerde data te bevragen. Voorbeelden hiervan zijn organisatiestructuren, materiaallijsten, documenthiërarchieën, wegbeschrijvingen, vluchtgegevens, ... In dit artikel gaan we kijken hoe recursieve SQL gebruikt kan worden om 'gaten' in een kolom te vinden. Deze kunnen bijvoorbeeld voorkomen in een *primary key*-kolom waar de waarden oplopend worden toegekend aan nieuwe rijen. Bij het weghalen van rijen vallen er dan 'gaten' in deze oplopende reeks. En dan kan het interessant zijn deze gaten terug op te vullen - met welke waarden?

Recursie is in DB2 geïmplementeerd door gebruik te maken van *common table expressions* (CTEs), die eveneens nieuw zijn in DB2 V8, en ook los van recursie zeer nuttig zijn. Een CTE kan gezien worden als een tijdelijke tabel (of eigenlijk een view) die gebruikt wordt tijdens de uitvoering van één SQL-statement. Deze tijdelijke view wordt aan het begin van de query gedefinieerd met een WITH-clause. Hierin krijgt de CTE een unieke naam die verder in de query gebruikt kan worden. Tevens staan hier de kolomnamen gedefinieerd die het resultaat zijn van de CTE. Het recursieve deel van het SQL-statement zit nu precies in deze CTE, doordat deze zichzelf oproept. Laten we dit eens van dichtbij bekijken aan de hand van voorbeeld 1.

*Voorbeeld 1*

```
WITH holetable (hole) AS
 ( select pno + 1
   from   persons
   where  pno + 1 NOT IN (select pno from persons)
  UNION ALL
   select hole + 1
   from   holetable
   where  hole < (select max(pno) from persons)
     and  hole + 1 NOT IN (select pno from persons)
 )
select hole
from   holetable
order by hole;
```

Een recursieve SQL is opgebouwd uit 3 delen:

1. initialisatie : in deze stap wordt de basis gelegd waarop de recursie zal voortbouwen. In ons voorbeeld is dit een query die van elk gat in de kolom *pno* van de tabel *persons* de eerste waarde gaat zoeken.

2. recursie: hier roept de CTE zichzelf terug op. In de FROM clause wordt er een select uitgevoerd op de CTE. Zolang het resultaat van de CTE hierdoor verandert, blijft deze zichzelf oproepen: er treedt met andere woorden recursie op.

De constructie met UNION ALL is verplicht. De eindconditie (*hole < max(pno)*) is uiteraard belangrijk om oneindige recursie te vermijden!

3. gebruik van de CTE: in de eigenlijke SELECT wordt de CTE ondervraagd, zodat de recursie daadwerkelijk plaatsvindt.

Als alternatief kan de CTE ook alle waarden tussen *min(pno)* en *max(pno)* genereren, waarvan nadien alle waarden die in de tabel *persons* voorkomen weer weggehaald worden in stap 3. Deze oplossing - zie voorbeeld 2 - is iets performanter.

*Voorbeeld 2*

```
WITH holetable (hole) AS
 ( select min(pno)
   from   persons
  UNION ALL
   select hole + 1
   from   holetable
   where  hole < (select max(pno) from persons)
 )
select hole
from   holetable
where  hole NOT IN (select pno from persons)
order by hole;
```

Een implementatie met NOT EXISTS i.p.v. NOT IN is uiteraard nog performanter - dit laten we als een oefening voor de lezer. We vernemen uiteraard graag alle interessante bevindingen die u bij het experimenteren met recursieve SQL bent tegengekomen.

Er kunnen uiteraard alternatieve oplossingen worden bedacht om dit probleem op te lossen, die ook in oudere versies van DB2 werken, bijvoorbeeld door programmacode te schrijven. Hieraan zijn echter ook nadelen verbonden zoals teveel data die worden opgehaald vanuit DB2 naar de applicatie, intermediaire tabellen die gegenereerd moeten worden, complexe programmalogica, ... Zeker bij hiërarchisch gestructureerde data is dit het geval.

Alhoewel recursie soms wordt beschouwd als te moeilijk en te inefficiënt, kan het dus toch een elegante oplossing zijn om zeer specifieke problemen op te lossen.

In een volgend nummer wordt ingegaan op het gebruik van recursie om tabellen te 'denormaliseren' of te 'roteren' (pivoting), of om meer flexibele kolomfuncties zelf te implementeren, zoals b.v. een 'concat' van alle velden binnen een groep.

# Visual Explain voor DB2 v8 op z/OS

*Eric Everaert*

*Visual Explain for DB2* voor z/OS laat toe om een grafische weegave te krijgen van het toegangspad dat DB2 heeft gekozen voor een SQL-instructie. Dankzij dergelijke grafische voorstellingen is het niet meer nodig om manueel de (niet al te leesbare) informatie in de PLAN_TABLE te interpreteren. Bovendien wordt het gemakkelijk om relaties tussen database-objecten te zien, met name tussen tabellen en indexen, maar ook om operaties te zien zoals een table scan, een index-toegang, of een table join. Al deze informatie, die de basis vormt voor het verbeteren van de performantie van queries, wordt zeer duidelijk geïllustreerd in grafieken.

Visual Explain kan ook gebruikt worden om gepersonaliseerde rapporten te genereren over SQL-instructies, om parameters van een subsysteem te visualiseren, evenals gegevens van de PLAN_TABLE, de kosttabel (DSN_STATEMNT_TABLE) en de functietabel.

## De grafische voorstelling van het toegangspad

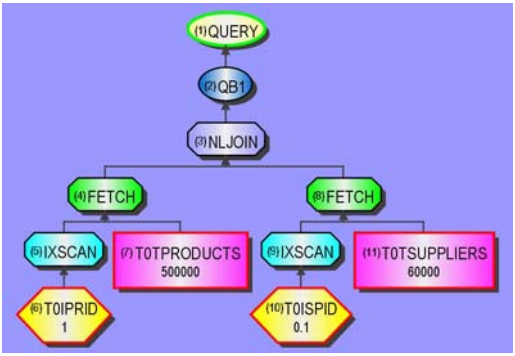
Visual Explain geeft een grafische voorstelling van het optimale toegangspad (*access path*) door DB2 bepaald tijdens de fase van de query optimisatie. Deze voorstelling laat duidelijk de opeenvolgende stappen zien waarmee DB2 de gegevens zal opvragen die tot het gevraagde resultaat van de gegeven query zal leiden.

Het grafische toegangspad bestaat uit *nodes* (blokjes) en *vertices* (connecties) ertussen. De nodes stellen gegevensbronnen (tabellen en indexen) voor, maar ook operatoren (zoals een join) en delen van de query (zoals een subquery). Elke node kan slechts één parent node hebben (behalve bij recursieve SQL), maar eventueel meerdere child nodes. Op die manier ontstaat dus een hiërarchische boomstructuur. De pijlen op de vertices tonen de richting aan van de datastroom en dus de volgorde van uitvoering van de operatoren. In het algemeen staan de nodes die data voorstellen (dus tabellen en indexen) onderaan in de grafiek; dat is waar de datatoegang begint.

Bepaalde operaties in een toegangspad, zoals joins en index-gebaseerde tabeltoegang, worden in de grafiek voorgesteld door een groep nodes, die *constructies* genoemd worden. Een groot deel van deze constructies starten met een definitie-node met daarin de naam van de operatie. In Figuur 1 zijn dit bijvoorbeeld de nodes NLJOIN die een 'nested loop join' voorstelt, de twee nodes IXSCAN die een toegang naar een index (only) voorstellen, en de twee nodes genaamd FETCH die een index-gestuurde tabeltoegang voorstellen. De volledige join bestaat uiteraard uit de twee deelbomen die uit de NLJOIN-node vertrekken, de IXSCAN-node verwijst naar de betreffende index-node, en een FETCH-node verwijst naar zowel de tabel als de

indexscan. De gezamenlijke nodes in een constructie bestaan uit alle gegevensbronnen en deeloperaties die voor de eigenlijke operatie nodig zijn.

*Figuur 1: Grafische voorstelling in Visual Explain van een join van twee tabellen*




## Nodes

Een grafiek van een toegangspad kan drie soorten nodes bevatten:

- Nodes van *gegevensbronnen*: stellen objecten voor van de database die gegevens bevatten. In de database worden deze objecten fysisch geïmplementeerd als een tabel of een index.
- *Operator*-nodes: deze corresponderen met een operatie zoals bijvoorbeeld een nested loop join die uitgevoerd wordt hetzij rechtstreeks op twee gegevensbronnen hetzij op het resultaat van een voorafgaande operator (zoals b.v. een index-toegang of een sortering).
- *Hulpnodes*: dit zijn alle andere nodes. In Figuur 1 gaat het om de node 'QB1' die een query block (meestal een subquery) voorstelt, en de node 'QUERY' die het SELECT-statement zelf voorstelt.

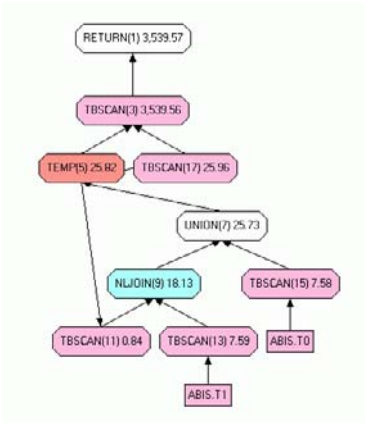
Het voorbeeld in Figuur 1 bevat nodes van elk van de drie soorten. Elke node kan nog bijkomende informatie bevatten, zoals b.v. de grootte van een tabel of de cluster ratio van een index.

## Meerdere 'Query Blocks'

Eén enkele query bestaat meestal uit meerdere subqueries. Een subquery is te herkennen aan het feit dat het woord SELECT (maar eventueel ook INSERT, UPDATE, DELETE of VALUES) verschijnt binnen een data manipulation statement. In dat geval wordt elk van deze subqueries in het grafische toegangspad voorgesteld door een *'query block*'. Dit geldt trouwens ook voor de samenstellende delen van een UNION, want ook daar komt het woord SELECT meerdere keren voor. Elke subquery kan op zijn beurt één of meerdere query blocks bevatten, meestal in de FROM-clause (nested table expression) of de WHERE-clause (nested query), maar ook een common table expressi-

on (CTE) is voor Visual Explain een query block. Een dergelijke 'query/subquery'-relatie wordt grafisch voorgesteld door een hiërarchische 'parent/child'-relatie in de vorm van een deelboom.

*Figuur 2: Grafische voorstelling in Visual Explain van recursieve SQL*




Wanneer een subquery verwijst naar minstens één kolom van zijn parent-query of eventueel zelfs van een query nog hogerop in de boomstructuur, spreekt men van een gecorreleerde subquery. Zo niet, dan spreekt men van een niet-gecorreleerde subquery. Deze laatste kan uitgevoerd worden op hetzelfde ogenblik als de hoogste erboven staande niet-gecorreleerde parent query in dezelfde hiërarchie. In Visual Explain wordt deze parent-query ook wel de '*parent subquery do-at-open*' genoemd, in termen van zijn relatie met de gegeven niet-gecorreleerde subquery. De uitvoering van een gecorreleerde subquery daarentegen is gelieerd aan die uitvoering van de parent-query. De manier waarop deze twee query blocks interageren, kan eveneens voorgesteld worden in de grafische hiërarchie van de voorstelling door Visual Explain.

## Activeren van Visual Explain

Het venster 'Enable Visual Explain' laat toe om Visual Explain te activeren voor het opgegeven subsysteem. Daarvoor moet er eerst een connectie gemaakt worden met het subsysteem via het paneel genaamd 'List Databases'.

Het activeren van Visual Explain impliceert het creëren (op z/OS) van een de benodigde werkomgeving om Visual Explain voor z/OS te kunnen gebruiken. Hiertoe worden alle nodige EXPLAIN-tabellen automatisch gecreëerd (of gevalideerd) op het opgegeven subsysteem. Het activeren van Visual Explain is slechts éénmalig nodig, weliswaar voor elk subsysteem en elke user-id afzonderlijk.

# Nieuw in DB2 v8 voor z/OS - een overzicht

*Peter Vanroose (ABIS)*

In vorige nummers van Exploring DB2 hebben we reeds verschillende nieuwe mogelijkheden van versie 8 onder de loep genomen: Unicode, multi-row inserts, multi-row fetches, MQTs, REOPT(ONCE), recursie, tabelgestuurde partitionering, grotere tablespaces en meer partities, schema-evolutie, ...

In dit artikel overlopen we kort de belangrijkste 'highlights' van versie 8, zeker voor wat betreft de dagdagelijkse praktijk, min of meer in volgorde van belangrijkheid.

Het valt vooral op dat er ingrijpende aanpassingen zijn - vergeleken met versie 7 - zowel op het vlak van de applicatieontwikkeling, als op het vlak van database-administratie, als op systeemniveau.

Verschillende ingrijpende aanpassingen hebben hun repercussies op elk van die niveau's. Denken we hierbij vooral aan de conversie van de catalog naar Unicode, en het (interne) gebruik van 64-bit adressering voor bijna alle componenten van DB2.

In het bijzonder geldt dit voor de werking van de optimizer. Die is enerzijds een stuk intelligenter geworden, waardoor veel access paths een stuk efficiënter zijn, maar is anderzijds ook gevoeliger geworden voor wat de statistieken in de catalog vertellen. Uiteraard heeft o.a. ook de 64-bit adressering een iets 'logger' systeem tot gevolg, wat echter in de meeste gevallen ruimschoots gecompenseerd wordt door de efficiëntere optimizer en de betere benutting van de system resources, vooral dan het intern geheugen.

Dus ook zonder expliciet gebruik te maken van de nieuwe mogelijkheden in versie 8, worden we geconfronteerd met een aantal verschillen, in het bijzonder dus met (automatisch) betere access paths voor onze queries en een betere (automatische) lock avoidance, i.h.b. in combinatie met utilities.

Er zijn echter enkele gevallen bekend waarbij een 'out-of-the-box' rebind van een package op versie 8, dus zonder gebruik te maken van de nieuwe mogelijkheden van versie 8, een performantie-degradatie gaf van 5% (en in compatibility mode zelfs 20%) t.o.v. versie 7. Het is dus van cruciaal belang, zeker voor dergelijke problematische queries, om nauwgezet performantie-onderzoek te doen, zowel vóór als na de transitie naar versie 8. Dit houdt onder andere in dat - in overleg tussen ontwikkelaar en DBA - access paths moeten opgevraagd worden met Explain, zowel in versie 7 als in versie 8, waarbij opvallende verschillen van naderbij moeten bekeken worden. Voor de enkele queries waarbij de geschatte kost een stuk hoger uitkomt dan voordien, dient herschrijven van SQL overwogen te worden. Gelukkig kan Visual Explain ons hierbij een beetje helpen.

Overlopen we nu enkele nieuwe features van versie 8, gezien vanuit het standpunt applicatieperformantie, in volgorde van belangrijkheid:

## Multi-row fetch

Voor een beschrijving van het 'hoe' en 'wat' verwijzen we naar 'Dossier 8', Exploring DB2, jaargang 2, nummer 4, december 2003. Kort samengevat laat DB2 nu toe om meerdere rijen in één enkele FETCH op te halen.

Een bestaande applicatie herschijven om deze mogelijkheid te beginnen gebruiken vereist weliswaar (ingrijpende) aanpassingen aan de programmatuur, met name arrays declareren en loops volledig herschrijven.

Maar het levert potentieel een performantiewinst op tot 20% (of uitzonderlijk zelfs tot 40%). Vooral wanneer een opgevraagde rij zeer kort is (b.v. enkel een SMALLINT) kan een grote tijdswinst verwacht worden.

Het aantal rijen dat gelijktijdig opgevraagd wordt, hoeft niet eens zo groot te zijn; meer dan 100 heeft trouwens weinig zin omdat de extra performantiewinst dan verwaarloosbaar wordt.

Denk in interactieve context in termen van 'een scherm vol', dus een 20-tal rijen. Op die manier kan eventueel zelfs een expliciete iteratie in de applicatie uitgespaard worden, en de benodigde array was waarschijnlijk toch al aanwezig. Dus als extra bonus: leesbaardere en beter onderhoudbare COBOL of PL/I.

## Access-path selectie

De versie 8-optimizer werkt fundamenteel anders dan in versie 7. Hij geeft in het algemeen een betere performantie, maar kan anderzijds, bij ondoordacht gebruik van REBIND, ook tot serieuze degradaties leiden.

Het is vooral belangrijk om te zorgen voor juiste en voldoende statistische gegevens. In het bijzonder de aanwezigheid van kolomstatistieken beïnvloedt de correctheid van de filterfactor berekeningen van de optimizer sterk, en deze hebben uiteraard een doorslaggevend effect op beslissingen over het gebruik van een index en de manier waarop, alsook over de gekozen join-volgorde en -methode. (In het bijzonder lijkt de hybrid join-methode populairder geworden te zijn bij de versie 8-optimizer...)

Zorg er dus voor dat op zijn minst voor kolommen die gebruikt worden in WHERE-condities in kritische applicaties, de RUNSTATS-parameter COLGROUP meegegeven wordt. Dit is des te belangrijker voor multicolom-indexen, zelfs (of misschien wel vooral) indien die WHERE-conditie host-variabelen bevat en uw statische SQL niet met REOPT(vars) wordt geBIND.

Opnieuw dus een aspect waarbij nauwe samenwerking tussen applicatieontwikkelaars en DBA cruciaal is. Zie ook 'Dossier 8' , Exploring DB2 jaargang 2, nummer 5, februari 2005.

Zorgen voor de juiste statistieken vóór een REBIND in versie 8 levert gegarandeerd performantiewinst op. Gemiddeld kan zo'n 5% winst verwacht worden, en dit zonder ook maar in het minst iets aan de SQL te veranderen.

## Non-matching data types

Een aantal predikaten die nog stage-2 waren in versie 7 zijn nu stage-1 (en indexeerbaar) in versie 8. I.h.b. gaat het over het vergelijken van datatypes zoals INT met DECIMAL of CHAR met VARCHAR. Ook dit kan mogelijk automatisch winst opleveren bij een REBIND, maar waarschijnlijk was uw applicatie al zodanig getuned dat dit soort predikaten toch al niet voorkwamen.

Bekijk hiervoor misschien nog eens 'Dossier 8' , Exploring DB2 jaargang 2, nummer 5, februari 2005. Deze ingrijpende aanpassing levert eveneens een betere communicatie met applicaties die in Java geschreven zijn: Java kent namelijk geen 'decimal' noch 'char(nn)', enkel 'float' en 'varchar' ...

## En verder ...

Indexen kunnen nu ook als NOT PADDED aangemaakt worden, waardoor ze nu ook de lengte van een VARCHAR-veld opslaan. Behalve het kleiner worden van de index-entries zelf, in het bijzonder wanneer de gemiddelde lengte van dat VARCHAR-veld een stuk kleiner is dan de maximale lengte, biedt deze nieuwe mogelijkheid ook perspectieven om een table scan (automatisch) om te toveren in een index-only (screening) access. Wanneer geen van beide voordelen verwacht worden voor een bepaalde index, kan die nog steeds als PADDED gedeclareerd worden.

De catalog wordt geconverteerd naar Unicode; terzelfdertijd wordt het een stuk makkelijker om data in verschillende encoding-schemes te combineren en door elkaar te gebruiken. z/OS helpt trouwens een handje bij de conversie tussen EBCDIC en Unicode indien die nodig zou zijn. Verder krijgen een aantal objecten (zoals b.v. routines) een encoding-scheme parameter.

Door het gebruik van 64-bit-adressering voor (bijna alle) virtual storage, vallen een aantal (oude) limieten weg. Zo zijn o.a. grotere buffer pools mogelijk, iets wat zeker in de context van read-only databases en data warehousing niet onbelangrijk is: meer data kan mogelijk langere tijd resident in het geheugen zitten, wat disk I/O aanzienlijk kan reduceren. Nu alleen nog hopen dat de aanwezige real storage verhoudingsgewijs mee groeit...

Tenslotte ook nog even wijzen op enkele gewijzigde defaults, vooral op het database-administratie-niveau, o.a. bij het creëren van tablespaces en enkele utilities. En het feit dat voor het eerst de catalog ingrijpender gewijzigd is dan alleen maar het toevoegen van kolommen: enkele zijn breder geworden of zijn van CHAR naar VARCHAR gepromoveerd. Let dus op wanneer uw applicaties queries draaien op de catalog. Er zijn er zelfs verdwenen: SYSIBM.SYSTABLES.CARD !

# DB2 en content management - 4: document flow

*Eric Venmans*

## Inleiding

In de vorige drie bijdragen over de DB2 Content Manager (DB2 ContMgr) werd aandacht besteed aan de algemene architectuur van het product, aan het onderliggende datamodel met zijn bestanddelen: items, componenten, objecten, attributen, relationships ..., en aan het gebruik vanuit applicaties, al dan niet via de ingebouwde zoektaal. In dit laatste deel zullen we kort het werken via 'document routing processen' behandelen.

## De workflow service

Bij het gebruiken van documenten kan men een aparte service aanroepen die wordt aangeduid als document routing. Het is een ingebouwde workflow service. In deze document flow spelen de reeds uitvoerig beschreven items en objecten een rol, maar komen ook nieuwe begrippen naar voor, zoals: processen, worklists, work nodes, collection points, work baskets ...

Een document routing process is een verzameling work nodes. Elke work node is een stap in dit process. Het kan twee vormen aannemen:

- collection point - een node waar een aantal items verzameld worden. Als de verzameling 'volledig' is, is een 'work package' gevormd; dit 'package' bevat alle afgesproken documenten of onderdelen ervan en dit in een afgesproken status;
- work basket - een queue met 'work packages'. Gebruikers kunnen verantwoordelijk zijn voor het verwerken van packages. Na verwerking worden betrokken items naar een volgend collection point verstuurd; welk 'collection point' kan men afhankelijk maken van het resultaat van de package verwerking.

Het verwerken van een 'work package' kan, zoals net aangegeven, worden uitgevoerd door een gebruiker. Dit kan gebeuren via een programma of een geprogrammeerd hulpmiddel (een tekstverwerker, een spreadsheet ...). De verwerking kan ook volledig autonoom worden uitgevoerd door een applicatie.

Voor het toekennen van packages aan applicaties en gebruikers, worden ACLs (Access Control Lists) gebruikt. Elke ACL (lijst met één of meerder gebruikers) wordt geassocieerd met 'document routing processen en met 'work baskets'. Op basis van deze associaties

krijgt een gebruiker een eigen 'worklist'. Hierop verschijnen die 'work packages' waarvoor hij of zij via het ACL-mechanisme geautoriseerd is. Eenzelfde package kan op de worklist van meerdere gebruikers terecht komen. De eerste gebruiker die zulk een package selecteert, haalt het weg uit de bijhorende 'work basket' waardoor het van de worklist van andere gebruikers verdwijnt. Na verwerking gaat het resultaat naar een volgende 'work node' afhankelijk van de definitie in het controlerende 'document routing process'. Het definieren van dergelijk process is vrij eenvoudig, maar tevens ook beperkt wat mogelijkheden betreft.

DB2 ContMgr kan worden geconfigureerd als een (afhankelijk) onderdeel van MQ Workflow Manager.

### Besluit

De automatisering van informatieverstrekking en -verwerking gaat steeds verder. De informatie zelf krijgt steeds meer exotische vormen. Om daarmee efficiënt om te gaan, zijn krachtige hulpmiddelen nodig. De DB2 ContMgr is er één van, en zeker niet de minste. Alleen zal het gebruik ervan moeten passen in een algehele bedrijfsstrategie i.v.m. het omgaan met informatie, zowel met interne als externe, met centraalbeheerde als met gedistribueerde!

## OVERZICHT VAN DE GEPUBLICEERDE ARTIKELS

| Artikel | Nummer |
|---|---|
| Over datum, tijd en tijdsduur in DB2 - 1 | 4-1 |
| Over datum, tijd en tijdsduur in DB2 - 2 | 4-2 |
| Tabel-partitionering in DB2 versie 8 | 4-4 |
| Gaten in tabellen: recursieve SQL in DB2 | 4-5 |
| Visual Explain voor DB2 v8 op z/OS | 4-5 |
| Nieuw in DB2 v8 voor z/OS - een overzicht | 4-5 |
| Tekstcodeerschema's en Unicode met DB2 v8 | 4-4 |
| Over MQSeries enablement in DB2 UDB v8 | 4-1 |
| RAD en DB2 | 4-3 |
| DB2 en content management - 1 | 4-2 |
| DB2 en content management - 2 - het data model | 4-3 |
| DB2 en content management - 3 - zoekfaciliteiten | 4-4 |
| DB2 en content management - 4 - document flow | 4-5 |
| Oracle? SQLServer? - het systeem (zie ook web versie) | 4-1 |
| SQL Server Architecture: DB - Filegroup - Extent (zie ook web versie) | 4-2 |
| Tabelimplementatie in DB2, Oracle en SQL Server (zie ook web versie) | 4-4 |

# CURSUSPLANNING JAN - JUL 2007

| | | |
|---|---|---|
| DB2 concepten | | op aanvraag |
| DB2 for OS/390, een totaaloverzicht | 1825 EUR | 05.02 (L), 19.02 (W), 16.04 (L), 10.05 (W), 18.06 (L), 16.07 (W) |
| DB2 UDB, een totaaloverzicht | 1750 EUR | 19.02 (W), 16.04 (L), 10.05 (W) |
| RDBMS concepten | 350 EUR | 05.02 (L), 19.02 (W), 16.04 (L), 10.05 (W), 18.06 (L), 16.07 (W) |
| Basiskennis SQL | 350 EUR | 06.02 (L), 20.02 (W), 17.04 (L), 11.05 (W), 19.06 (L), 17.07 (W) |
| DB2 for OS/390 basiscursus | 1125 EUR | 07.02 (L), 21.02 (W), 18.04 (L), 14.05 (W), 20.06 (L), 18.07 (W) |
| DB2 UDB basiscursus | 1050 EUR | 21.02 (W), 18.04 (L), 14.05 (W) |
| SQL workshop | 750 EUR | 15.02 (L), 18.03 (W), 26.04 (L), 29.05 (W), 28.06 (L) |
| Extended SQL in DB2 | 425 EUR | 02.03 (L), 13.04 (W) |
| Gebruik van DB2 procedural extensions | 425 EUR | 18.01 (W), 25.05 (L) |
| DB2 for OS/390 programmering voor gevorderden | 800 EUR | 15.02 (L), 29.05 (W) |
| DB2 for OS/390: SQL performance | 1275 EUR | 12.02 (L), 20.06 (W) |
| XML in DB2 | | op aanvraag |
| DB2 for OS/390 database administratie | 1700 EUR | 13.03 (W), 21.05 (L) |
| DB2 for z/OS in een Java omgeving | 425 EUR | 10.05 (W), 14.06 (L) |
| DB2 for z/OS operations and recovery | 1425 EUR | 07.03 (UK), 03.04 (W) |
| DB2 for z/OS systems performance and tuning | 1000 EUR | 26.02 (UK), 23.04 (UK), 24.05 (L), 25.06 (UK) |
| DB2 UDB DBA 1 - Kernvaardigheden | 1700 EUR | 26.03 (L), 29.05 (W) |
| DB2 UDB DBA 2 - configure & tune | 1275 EUR | 12.02 (L), 26.06 (W) |
| DB2 UDB v8 Upgrade | | op aanvraag |

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