ABIS NIEUWSBRIEF VOOR DB2 PROFESSIONALS

JAARGANG 4 - NUMMER 1, SEPTEMBER 2005

OPEN CURSOR

We leggen de laatste hand aan dit nummer van Exploring DB2 op het moment dat Larry Ellison ons meedeelt dat hij zopas aanvaard heeft Siebel op te kopen - samen met Peoplesoft/JDEdwards en Oracle Apps is Oracle nu de nummer een in de CRM markt. Wat een toeval dat we juist in dit nummer stilstaan bij alternatieve relationele databases. Want wie van ons komt niet in aanraking met Oracle, SQL Server? We bekijken beide alternatieven door een DB2 bril! Het begin van een nieuwe reeks.
En natuurlijk hebben we ook oog voor de DB2 applicatieontwikelaar. En speelt integratie een rol.

Wie weet - misschien treffen we mekaar volgende week op de Oracle Open World Conference!

Het ABIS DB2-team.

IN DIT NUMMER:

- Steeds meer datummanipulaties worden door DB2 ondersteund - een opfrissing, in *Over datum, tijd en tijdsduur in DB2*.
- Naast DB2 hebben we allemaal ook andere relationele databases in huis. Wat zijn nu de verschillen - *Oracle? SQLServer? - het systeem*. We beginnen een nieuwe reeks!
- We hadden het reeds over MQSeries en DB2 - wat is er nieuw? Lees *Over MQSeries enablement in DB2 UDB v8*.
- *Cursusplanning augustus 2005 - november 2005*.

CLOSE CURSOR

In een volgend nummer staan we stil bij de fysieke structuur van Oracle en SQLserver. Verder behandelen we de DBA features die WebSphere Application Developer u kan bieden in een DB2 achtergrond.

Tot dan!

Over datum, tijd en tijdsduur in DB2

Peter Vanroose (ABIS)

DB2 (en eigenlijk SQL in het algemeen) heeft drie standaard datatypes voor het voorstellen van datums en tijdstippen:

- DATE, voor datums, d.w.z., dag, maand en jaar;
- TIME, voor tijdstippen binnen een dag: uur, minuten en seconden;
- TIMESTAMP, voor een combinatie van DATE, TIME, en miljoensten van een seconde.

Deze drie datatypes zijn standaard SQL en worden ondersteund door de meeste relationele databases (DB2, Oracle, SQL Server, etc.).

Toch zijn er een aantal aspecten i.v.m. deze datatypes die minder 'standaard' zijn dan we zouden wensen, of zelfs afwijkend zijn overheen verschillende versies van DB2. In deze bijdrage worden enkele van deze aspecten kort belicht, en worden pasklare oplossingen geboden voor enkele veelgestelde vragen.

Tijdsduur

Naast de drie opgesomde datatypes, ondersteunen veel relationele databases ook afzonderlijke datatypes voor het voorstellen van een tijdsduur of een tijdsverschil. Zo gebruikt Oracle bijvoorbeeld INTERVAL datatypes. DB2 doet dit niet.

Voor datums is dit niet echt een probleem: het verschil tussen twee datums is een positief of negatief geheel getal van het type INT, nl. het aantal dagen 'tussen' beide. In de meeste, zoniet alle gevallen moet de DAYS functie worden gebruikt om de conversie van datum naar 'dagnummer' mogelijk te maken. Zoniet, is het resultaat van betrokken bewerking vaak 'verrassend'...

Omgekeerd is het eveneens mogelijk een aantal dagen bij een datum op te tellen of af te trekken; DB2 eist hiervoor wel het gebruik van zogenaamde LABELED DURATIONS: DAYS, MONTHS, YEARS, jammer genoeg niet WEEKS. Verschillende LABELED DURATIONS kunnen worden gecombineerd in één expressie - maar let op de volgorde! Het resultaat van deze expressies (resulterende jaartal) moet tussen 1 en 9999 liggen.

Voorbeeld 1: Tijdsduur - DATE

```
DATE('2005-12-11') - '2005-12-10'          -->  1
DAYS('2005-12-10') - DAYS('2005-12-11')    --> -1
DATE('2005-11-09') + 2 MONTHS + 22 DAYS    --> 31.01.2006
DATE('2005-11-09') + 22 DAYS + 2 MONTHS    --> 01.02.2006
CURRENT DATE - 3000 YEARS                  --> ongeldig
```

Op eenzelfde wijze kan ook met TIME en TIMESTAMP omgesprongen worden. Inderdaad, het verschil tussen twee TIME of TIMESTAMP waarden is een getal dat het 'gecodeerde' verschil geeft tussen die twee tijdstippen. Omdat tijdsverschillen gewoon numerieke, onbenoemde grootheden zijn, kunnen ze eventueel ook weer in expressies gebruikt worden. Specifieke LABELED DURATIONS kunnen eveneens worden gebruikt (HOURS, MINUTES, SECONDS). Merk op:

- bij een TIMESTAMP kunnen (gehele) dagen, maanden of jaren bijgeteld of afgetrokken worden; bij een TIME niet;
- het resultaat van de bewerking kan positief of negatief zijn;
- de output van een TIMESTAMP bewerking dient als volgt geïnterpreteerd: yyyymmddHHMMSS,xxxxxx; het resultaat van een TIME bewerking is steeds HHMMSS;
- het domein van TIME omvat alle waarden van 00:00:00 tot 23:59:59, en werkt dus 'modulo 24 uur' bij overschrijden van middernacht. Dit is echter niet het geval bij TIMESTAMP.

Voorbeeld 2: Tijdsduur - TIME

```
current timestamp -
(current timestamp - 1 day - 2 hours - 3 minutes - 5 seconds)
    --> 1020305.000000

current time - (current time - 1 hour - 3 minutes - 5 seconds)
    --> 10305

time('22:00:00') + 200 minutes - '22:00:00'
    --> -20400

timestamp('22:00:00') + 200 minutes - timestamp('22:00:00')
    --> 32000
```

Uitdagingen

In tegenstelling tot de numerieke datatypes en die voor tekst, bestaat er voor de datum- en tijdtypes geen 'eigen' voorstellingswijze in SQL of in een 'host-language': ingevoerde datum- en tijdconstanten moeten als tekstvelden voorgesteld worden in SQL, en DB2 zorgt voor de nodige conversie, echter niet steeds automatisch.

Hier duikt al een eerste mogelijkheid tot verwarring op: is '12/11/2005' nu 12 november dan wel 11 december? Of is de interpretatie afhankelijk van de lokale configuratie van DB2?

Gelukkig is dit laatste niet het geval, maar om alvast deze vorm van verwarring te vermijden, gewoon al voor de leesbaarheid en de uitwisselbaarheid met b.v. andere software, is het toch wel aan te raden om voor datumconstanten de ISO-notatie te gebruiken: '2005-12-11' is ontegensprekelijk 11 december 2005.

Omgekeerd stelt zich een gelijkaardig probleem: in welke vorm zal DB2 de waarden in een DATE-kolom laten zien? Hiervoor is DB2 wel configuratie-afhankelijk, wat mogelijk tot interpretatieproblemen

kan leiden wanneer data uitgewisseld wordt over de landsgrenzen heen. Een goede gewoonte kan hier zijn om steeds expliciet te converteren m.b.v. de functie CHAR. Dit is vooral belangrijk bij het schrijven van SQL-scripts die wereldwijd gebruikt zullen worden en overal gegarandeerd dezelfde output moeten opleveren. Maar zelfs als uw script op de PC van uw collega's moet draaien, kan dit een goed idee zijn, want wie weet gebruiken jullie verschillende LOCALE instellingen ...

Een ideaal moment om het even te hebben over automatische conversie van geldige datumconstanten als '2005-12-11' naar type DATE.

- Indien het voor DB2 vaststaat dat een DATE (of TIME, TIMESTAMP) type vereist is, wordt de conversie automatisch uitgevoerd - in WHERE condities, als argument van een functie, als rechterlid in tijdverschuivingen, etc.

- Indien het voor DB2 niet duidelijk is welk datatype vereist is, moet de conversie expliciet worden aangevraagd - bijvoorbeeld als argument voor functies die meerdere datatypes toelaten, of in een stored procedure, of als linkerlid in tijdverschuivingen. Voor argumenten van user-defined functies of stored procedures is dit nogal omslachtig, en weinig gebruiksvriendelijk. Gelukkig is er een relatief eenvoudige manier om te zorgen dat een functie zowel DATE-argumenten als DATE-constanten aanvaardt - functie overloading.

Overloading? Zie *Over overloading...* op p. 13

Specifieke voorbeelden - toepassingen

In wat volgt worden tenslotte een aantal specifieke datum gerelateerde voorbeelden en uitdagingen aangehaald.

a. next_businessday

Een typische situatie waarbij, op basis van de 'betekenis' van een datum, een SQL-query verschillend moet werken, is wanneer men onderscheid wil maken tussen een weekdag/werkdag en een weekend, b.v. in een functie *next_businessday*, die na een vrijdag de volgende maandag moet geven.

Om voor een datum te achterhalen op welke dag van de week die valt, moeten we de ANSI SQL-functie DAYOFWEEK gebruiken. Standaard ANSI SQL, geïmplementeerd door DB2, beschouwt zondag als dag 1. Volgens ISO normen is maandag dag 1; gebruik hiertoe de functie DAYOFWEEK_ISO.

Indien gewenst kan het dagnummer met een CASE constructie in een naam worden omgezet.

Voorbeeld 3: next_businessday

```
CREATE FUNCTION next_businessday (t DATE)
RETURNS DATE
RETURN ( t + (CASE DAYOFWEEK(t)
               WHEN 6 THEN 3
               WHEN 7 THEN 2
               ELSE 1 END) DAY )
```

De implementatie van *next_businessday* kan het meest efficiënt met behulp van de CASE-instructie - zie voorbeeld 3.

b. Laatste dag van de maand?

Zeer vaak is het nuttig een onderscheid te maken tussen de laatste dag van de maand, en de andere dagen van diezelfde maand. Is er een eenvoudige functie om, gegeven een DATE-veld, te weten te komen of dit de laatste van de maand is?

Op DB2 voor z/OS kunnen we de functie LAST_DAY gebruiken; op andere platformen moet e.e.a. via een functie geïmplementeerd worden (of kan op een eenvoudige wijze in SQL statements worden opgenomen) - zie voorbeeld 4.

Voorbeeld 4: de laatste dag van de maand

```
<ofwel>
select ... from ...
WHERE DAY(my_date + 1 DAY) = 1

<ofwel generieker>
CREATE FUNCTION last_day (my_date DATE)
RETURNS DATE
RETURN (my_date - (DAY(my_date)-1) DAYS + 1 MONTH - 1 DAY)
```

c. Groeperen op datum

Vaak is het nuttig rapporten samen te stellen op basis van weekdag en/of maand-groepen. Strikt genomen hoeft dit geen complexe materie te zijn - een GROUP BY functie op basis van een typisch scalaire functie MONTH, DAYOFWEEK, ... zou voldoende moeten zijn. Alhoewel ondersteund door een aantal database-leveranciers, voorziet DB2 voor z/OS deze mogelijkheid pas vanaf DB2 versie 8. Zie voorbeeld 5 voor mogelijke alternatieven.

Een nadeel van eerder typisch en voor de hand liggende oplossingen als geschetst in voorbeeld 5 (a) en (b), is dat een dag van de week zonder rijen niet zal voorkomen in het eindresultaat; terwijl we bijvoorbeeld in dat geval een resultaatrij met COUNT= 0 willen zien.

Voorbeeld 5 (c) biedt ook hiervoor een - zeer efficiënt! - alternatief.

Voorbeeld 5: groeperen op datum - respectievelijk a, b en c

```
SELECT DAYOFWEEK(my_date) AS dvweek, COUNT(*) AS aantal
FROM my_table
GROUP BY DAYOFWEEK(my_date)

SELECT dvweek, COUNT(*) AS aantal
FROM (SELECT DAYOFWEEK(my_date) AS dvweek
      FROM my_table) AS t
GROUP BY dvweek

SELECT SUM(CASE DAYOFWEEK(my_date) WHEN 1 THEN 1 ELSE 0 END) AS zo,
       SUM(CASE DAYOFWEEK(my_date) WHEN 2 THEN 1 ELSE 0 END) AS ma,
       ...
       SUM(CASE DAYOFWEEK(my_date) WHEN 7 THEN 1 ELSE 0 END) AS za
FROM my_table
```

d. Vorige week/maand/kwartaal/...

Een veel voorkomende query bestaat erin, enkel de gegevens van een vorige periode (vorige maand, vorig kwartaal of vorig jaar ...) te lijsten. Waarbij men het natuurlijk NIET heeft over de 'elapsed' maand (bijvoorbeeld), maar over de kalendermaand. Zie voorbeeld 6 voor details.

Voorbeeld 6: de vorige periode

```
<traditioneel - elapsed maand>
select ... from ...
WHERE my_date > CURRENT DATE - 1 MONTH

<beter - vorige kalendermaand>
select ... from ...
... WHERE 12*YEAR(my_date)+MONTH(my_date) + 1 =
          12*YEAR(CURRENT DATE) + MONTH(CURRENT DATE)

<beter - vorig kwartaal>
select ... from ...
... WHERE (YEAR(CURRENT DATE)-YEAR(my_date)) * 4 +
    FLOOR((MONTH(CURRENT DATE)-1)/3) - FLOOR((MONTH(my_date)-1)/3) = 1
```

Natuurlijk is er een efficiënter alternatief - bijvoorbeeld een alternatief zonder gebruik te maken van scalaire functies als MONTH. Stuur ons uw alternatief, en u wordt met naam en toenaam gepubliceerd in het volgende nummer van Exploring DB2. Stuur uw oplossing naar training@abis.be!

Performance-issues bij gebruik van DATE en TIME

Een eenvoudige richtlijn: vermijd zoveel mogelijk het gebruik van datummanipulaties (zoals "+ n DAYS" of scalaire functies YEAR, MONTH en DAY) van velden in WHERE-condities die op grote tabellen worden toegepast.

Vaak zijn dergelijke condities te vervangen door BETWEEN, of door die functies enkel op constante waarden (zoals CURRENT DATE) toe te passen, die bijgevolg maar één keer hoeven berekend te worden voor de hele query, en niet telkens opnieuw voor elke rij uit de basistabel/het tussenresultaat.

Er zijn gevallen bekend waarbij de query-tijd door dit soort eenvoudige ingrepen teruggebracht werd van enkele uren tot enkele seconden ...

Oracle? SQLServer? - het systeem

Eric Everaert (ABIS)

Reeds 3 jaargangen lang hebben we in deze publicatie onze aandacht gericht op DB2, voornamelijk DB2 voor z/OS. We hopen - en we weten op basis van ontvangen feedback - dat u de in deze publicatie opgenomen info apprecieert. En alhoewel Exploring DB2 natuurlijk voornamelijk over DB2 zal blijven gaan, hebben we toch ook besloten een aantal niet-DB2-gerelateerde onderwerpen in deze publicatie te behandelen. Inderdaad, niet-DB2-gerelateerd, maar wel bekeken door een DB2-bril. Want hoeveel sites zijn vandaag de dag exclusief DB2-gericht? Hoeveel sites hebben er immers geen Oracle en/of SQL Server in huis?
Onze bedoeling is dus niet de aandacht af te leiden van DB2, noch van onze voorkeur en/of afkeur te laten blijken voor deze of gene RDBMS. Onze bedoeling is gewoon realist te zijn: wat moet u als DB2-specialist weten van andere database systemen waarmee u in aanraking komt?
Bedoeling is op een objectieve en gestructureerde wijze een aantal onderwerpen te behandelen, die DB2, Oracle en SQL Server aanbelangen. Natuurlijk gaan we er vanuit dat we u over DB2 niets meer moeten uitleggen. Onze aandacht zal gaan naar Oracle en SQL Server. Alternerend zal één van beide in de publicatie op papier dan wel op onze website worden behandeld.
Welke onderwerpen we willen behandelen? Onderwerpen die we als DB2-, Oracle- en SQL Server-experts als belangrijk ervaren voor DB2-experts. Maar we hopen ook onderwerpen te behandelen die u nauw aan het hart liggen; onderwerpen waarmee u in contact komt. Laat ons uw wensen weten op training@abis.be en in één van de volgende nummers van Exploring DB2 wordt dat onderwerp aangekaart. Gegarandeerd!

Eric Everaert, DB2 for z/OS, Oracle, SQL Server (ABIS)
Kris Van Thillo, DB2 for z/OS, DB2 UDB for Linux, Unix & Windows, Oracle (ABIS)

DB2 for z/OS en DB2 UDB (for LUW)

Wanneer we in dit artikel een onderscheid willen maken tussen de DB2-implementaties zullen we enerzijds spreken van DB2 for z/OS en anderzijds van DB2 UDB, waarbij we dan de DB2 UDB versies voor LUW (Linux, Unix & Windows) bedoelen.

Het Oracle systeem - de instance

Net zoals bij DB2 UDB spreekt men bij Oracle van een systeem, een INSTANCE. Concreet, de verzameling van processen en geheugen die ten dienste staan van een Oracle database, d.i. de verzameling van gegevens gemanipuleerd door de gebruiker. Een INSTANCE wordt geïdentificeerd op basis van een SID (System Identifier) en geconfigureerd aan de hand van een INSTANCE configuratie-file.

Het SQL Server systeem? zie *Het SQL Server systeem - de instance* op p. 14

Belangrijk is echter dat met één Oracle INSTANCE slechts één Oracle database kan worden geassocieerd. Deze database-structuur omvat o.a. transactielogs, een omgeving voor tijdelijke (sorteer)resultaten, een cataloog.

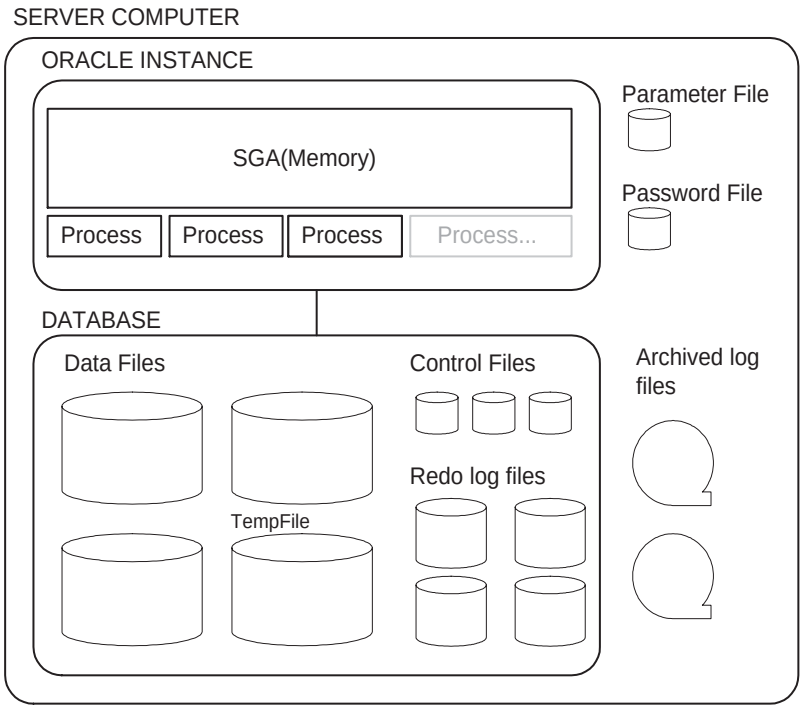
Het is nuttig dit even te vergelijken met DB2 for z/OS. Hier bevat een instance (subsysteem) verschillende databases, maar deze databases hebben geen concrete fysieke representatie: logs, cataloog, etc. bevinden zich op het niveau van de instance. En als we vergelijken met DB2 UDB, dan valt op dat DB2 UDB net zoals Oracle transactielogs, cataloog etc. organiseert op database niveau, maar dat DB2 UDB per instance wel meerdere databases toelaat (of deze opzet veelvuldig wordt gebruikt, is evenwel een andere zaak...).

Wat geheugenallocatie betreft, gedraagt Oracle zich misschien meer zoals DB2 for z/OS. Allocatie geschiedt op het moment dat de instance wordt gestart: bufferpools, redo log buffer, caches allerhande. Het gebruik wordt uitgesteld tot het moment van het openen van de enige database die met de instance kan worden geassocieerd.

En ook wat 'high availability' betreft vertoont Oracle 'DB2 for z/OS' karakteristieken: een database wordt geassocieerd met verschillende instances op verschillende nodes, het parallel sysplex idee dus. Van database partitioning als bij DB2 UDB is hier dus geen sprake!

In figuur 1 schetsen we schematisch een Oracle database-omgeving.

Figuur 1: een Oracle instance



De bovenstaande grafiek bevat een aantal elementen die in hoofdlijnen even moeten worden toegelicht. In detail wordt e.e.a. in volgende nummers uiteengezet.

- parameter file init<SID>.ora of spfile<SID>.ora: is de belangrijkste systeemconfiguratiefile, waarin o.a. de configuratie van geheugen, processen, enz. wordt opgeslagen. Inhoud wordt typisch aangepast door de Oracle systeembeheerder, gebruik makende van een GUI, dan wel rechtsreeks aan de hand van API's. Dit is conceptueel vergelijkbaar met DSNZPARM en/of SQLDBDIR en SQLDBCON.

- paswoord file: geëncrypteerd bestand met userid's, paswoorden en rechten, die de administrator moet toelaten administratieve taken uit te voeren over een niet-secure netwerk. Denk hierbij bijvoorbeeld aan het stoppen en starten van een Oracle-omgeving overheen een C/S-connectie.

- control file: file die de logische en fysieke structuur van de Oracle database beschrijft. Staat in voor de integriteit van de gegevens die zich in de database bevinden; belangrijk in de context van tijdssynchronisatie, log- en transactiemanagement, etc. Dit is conceptueel vergelijkbaar met DBDs en SYSLOGRANGE en/of SQLDBDIR en SQLDBCON.

- redo log file: bevat de Oracle transactielog, vergelijkbaar met de transactielogs op DB2-platformen. Het is de basis voor de creatie van 'archives'. Maar de file is ook cruciaal in Oracle-context voor een hele reeks taken die in een DB2-omgeving niet relevant zijn of op een verschillende manier worden ingevuld!

- data file: de containers met data, dit is de fysieke structuren waarin rijen als records worden opgeslagen. Ze zijn geassocieerd met tablespaces en indexspaces en zijn conceptueel identiek aan de DB2-omgeving. Een Oracle tablespace staat dichter bij een DB2 UDB tablespace dan bij een DB2 for z/OS tablespace-structuur.

- temp file: tijdelijke sorteerresultaten worden, indien nodig, in deze structuur opgeslagen. Vergelijk met de temporary tablespace/ database op respectieve DB2 platformen.

Inderdaad, voldoende verschillen en gelijkenissen om in vervolgartikels stap voor stap bepaalde onderwerpen verder uit te diepen!

Over MQSeries enablement in DB2 UDB v8

Kris Van Thillo (ABIS)

Reeds in een vorige editie van Exploring DB2 stonden we stil bij de problematiek van de integratie DB2 UDB for LUW (Linux, Unix & Windows) en MQSeries - zie Exploring DB2, Jaargang 3, nummer 1, 'DB2 en MQSeries - Integratie'. In dit artikel gaan we in op een tweetal nieuwe mogelijkheden, geïntroduceerd in DB2 UDB versie 8.

Transacties

Het transactieconcept is cruciaal in elke relationele database, zo ook in DB2. Het is immers de eenheid waarmee wijzigingen worden aanvaard (COMMIT) of geweigerd (ROLLBACK); we kennen immers allemaal de ACID-test!

In vorige versies van DB2 waren MQSeries-functies 'niet transactioneel'. Concreet: een rollback van een MQ-operaties gegenereerd via een DB2 Unit-of-work werd gewoon genegeerd. Boodschappen werden dus bijvoorbeeld NIET van een queue gehaald. De synchronisatie van DB2 en MQSeries is niet geïntegreerd; elk systeem is immers verantwoordelijk voor zijn eigen 'omgeving'.

Tijdens het activeren van MQ-support in DB2 UDB versie 8 is het nu mogelijk transactiesupport aan te vragen; een reeks nieuwe UDFs (User Defined Functions) worden aangemaakt, in een nieuw database schema db2mq1c. Het is tevens mogelijk bijvoorbeeld enkel UDFs met transactiesupport aan te maken.

Op dit moment wordt 'Single Phase Commit' ondersteund. Concreet heeft dit als gevolg dat:

- applicaties die via MQ UDFs een boodschap verzenden of ontvangen, deze boodschap maar als verzonder of ontvangen zullen beschouwen op het moment van een commit;
- fouten die optreden bij het uitvoeren van transacties die zend/ontvangst UDFs bevatten, ook al hebben deze fouten met het zenden/ontvangen op zich niets te maken, zullen aanleiding geven tot het ongedaan maken van de zend en/of ontvangst acties;
- verzonden boodschappen aan de hand van MQSeries UDFs binnen DB2 UDB zelfs pas ter beschikking komen van ontvangende DB2 UDFs na een COMMIT van de zendende applicatie.

Op dit moment kunnen DB2 MQSeries UDFs nog niet onbeperkt in alle DB2 statements worden opgenomen. Voorts wordt enkel 'Remote Unit of Work', of 'Distributed Unit of Work' met read only access naar alle databases behalve één, op dit moment ondersteund.

Asynchrone boodschappen en DB2

DB2 UDB versie 8 Information Integrator bevat een asynchrone MQ-Listener. Bedoeling is DB2-acties te activeren, op basis van boodschappen aan een MQSeries queue toegevoegd. De architectuur is zeer eenvoudig:

- een extern MQListener-proces monitort WebSphere MQ Message queues, opgegeven tijdens setup en configuratie van dat proces;
- als op de betreffende queue een boodschap wordt geplaatst, wordt een DB2 stored procedure opgestart, met de boodschap als input-argument;
- eventueel kan een reply op een queue worden geplaatst door dit proces; deze reply is typisch 'het resultaat' van de aangeroepen stored procedure.

Een aantal bijkomende elementen mogen niet aan onze aandacht ontsnappen.

▪ het is mogelijk de stored procedure actie te integreren met de lees- en schrijfacties door het listener proces geïnitieerd als één transactie. Default is dit echter niet het geval. Opzet van deze coördinatie vereist configuratie op het niveau van MQSeries.

▪ de stored procedure interface moet wat betreft input en output argumenten aan een aantal specifieke voorwaarden voldoen - uiteraard niet onlogisch;

▪ de MQListener vereist het bestaan van een DB2-configuratietabel, met gegevens omtrent de configuratie. Traditioneel moeten natuurlijk ook een aantal packages worden gebinded.

▪ onderscheid wordt gemaakt tussen een RUN user en een CONFIGURATIE user - deze laatste wordt door MQListener gebruikt om uit de configuratietabel configuratiegegevens op te halen; rechten om deze tabel te benaderen (via packages) zijn dus vereist. De RUN user hoeft enkel over de rechten te beschikken om aan te loggen aan de database en de stored procedure uit te voeren.

Deze technologie vult een nog ontbrekend stuk van de puzzel in: waar vroeger DB2 in staat was MQSeries-acties te starten (plaatst een boodschap op een queue waaraan een trigger-monitor is gehangen) is nu MQSeries op een eenvoudige manier in staat een DB2-actie te activeren. Deze actie kan veel omvatten; we denken bijvoorbeeld aan het 'on-line' opzoeken van gegevens, dan wel het voorbereiden voor batch-verwerking in een later stadium.

CURSUSPLANNING SEP - DEC 2005

DB2 concepten	375 EUR	26/09 (L), 16/11 (W)
DB2 for OS/390, een totaaloverzicht	1700 EUR	05-09/09 (W), 19-23/09(L), 17-21/10 (W), 28/11-01/12 (W)
DB2 UDB, een totaaloverzicht	1625 EUR	17-26/10 (W), 28/11-02/12 (W)
RDBMS concepten	325 EUR	05/09 (W), 19/09 (L), 17/10 (W), 21/11 (L), 28/11 (W)
Basiskennis SQL	325 EUR	06/09 (W), 20/09 (L), 18/10 (W), 22/11 (L), 29/11 (W)
DB2 for OS/390 basiscursus	1050 EUR	07-09/09 (W), 21-23/09 (L), 19-21/10 (W), 30/11-02/12 (W)
DB2 UDB basiscursus	975 EUR	24-26/10 (W), 30/11-02/12 (W)
SQL workshop	700 EUR	29-30/09 (W), 10-11/10 (L), 14-15/11 (W)
Extended SQL in DB2	400 EUR	17/10 (L)
Gebruik van DB2 procedural extensions	400 EUR	18/10 (L)
DB2 for OS/390 programmering voor gevorderden	750 EUR	19-20/09 (W). 17-18/11 (L)
DB2 for OS/390: SQL performance	1200 EUR	12-14/10 (L)
Database applicaties bouwen met Java	2800 EUR	19-28/10 (L), 18-30/11 (W)
XML in DB2	400 EUR	30/09 (W)
DB2 Content Manager	200 EUR	21/10 pm (L), 15/11 pm (W)
WebSphere Integrator	200 EUR	20/10 pm (L), 18/11 pm (W)
DB2 for OS/390 database administratie	1600 EUR	03-06/10 (W)
DB2 for z/OS in een Java omgeving	400 EUR	07/10 (W)
DB2 for OS/390 operations and recovery	1500 EUR	07-09/11 (W)

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

Over overloading...

Wat is functie overloading? En hoe helpt het ons in de context van de boven aangehaalde datumproblematiek?

Functie overloading komt neer op het definiëren van twee (of meer) functies met dezelfde naam, maar met verschillende aantallen argumenten en/of (in ons geval) verschillende datatypes voor de argument(en). DB2 bepaalt dan zelf, op basis van het aantal opgenomen argumenten en de actuele datatypes van de meegegeven argument(en) bij een functie-oproep, welk van de functies uitgevoerd wordt. Een voorbeeld illustreert eea. duidelijk.

Voorbeeld

```
CREATE FUNCTION next_day (t DATE)
RETURNS DATE
RETURN ( t + 1 DAY )

CREATE FUNCTION next_day (t VARCHAR(10))
RETURNS DATE
RETURN ( CAST(t AS DATE) + 1 DAY )
```

Dit geeft geen conflicten; inspectie van SYSIBM.SYSROUTINES (of van SYSCAT.FUNCTIONS) leert dat beide functies nu bestaan, allebei met naam 'NEXT_DAY', maar - en dat is cruciaal - met een verschillende FUNCTION SPECIFIC naam, bijvoorbeeld SQL0509301200000000 Deze naam garandeert uniciteit door de aanwezigheid van de creatie-timestamp in de naam. Deze naam vinden we terug in de kolom SPECIFIC of SPECIFICNAME (afhankelijk van de DB2-versie) van boven genoemde cataloogtabellen.

Bemerk:

- het gebruik van datatype VARCHAR voor het argument: als we CHAR(10) hadden gebruikt, dan zou DB2 die functie niet terugvinden voor het argument '2005-09-30'!
- het droppen van overloaded functies vergt het gebruik van een speciale DROP FUNCTION syntax - DROP SPECIFIC FUNCTION. Specificatie van de functie naam is in deze inderdaad onvoldoende éénduidig!

Het SQL Server systeem - de instance

Instance: één, twee, ...

Met SQL Server 2000 is het mogelijk om verschillende "instances" op eenzelfde server te draaien. Dit was niet het geval met SQL Server 7. Met die versie had men slechts één enkele instance ter beschikking en de naam van de NT-server volstond om er verbinding mee te maken. Het is nog steeds mogelijk om op deze manier te werken. Tijdens de installatie van SQL-server wordt er gevraagd of er een 'default' instance moet worden aangemaakt. Uiteraard kan men ook voor de nieuwe mogelijkheid opteren en meerdere instances kiezen en benoemen. Dank zij deze naam is het mogelijk om de verschillende instances te onderscheiden en er meerdere aan te maken op dezelfde server.

Elke instance wordt beschouwd als een andere “SQL server” server en er bestaat oorspronkelijk geen enkel verband tussen de verschillende instances op dezelfde machine. Uiteraard is het wel mogelijk om verbindingen aan te maken tussen de verschillende instances.

Om een nieuwe instance aan te maken, volstaat het om de installatie op te starten met de CD-ROM en te kiezen voor de installatie van een nieuwe instance i.p.v. een volledige installatie. Dit duurt slechts enkele minuten.

Elke instance is dan geïmplementeerd als een service, net zoals in DB2 UDB for LUW (Linux, Unix and Windows), en kan worden gestart en gestopt als gelijk welke Windows service. Er bestaat eveneens een kleine toepassing, de "SQL Server Service Manager" waar ook instances mee kunnen worden gestopt en gestart. Deze heeft bovendien het voordeel dat dit vanop afstand kan gebeuren.

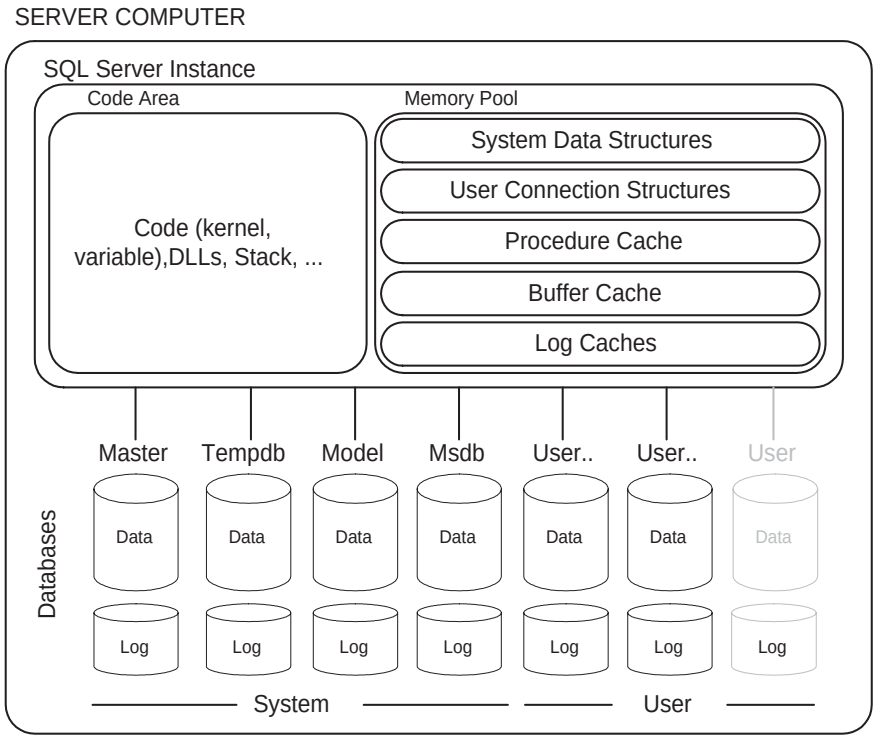
Het gebruik van verschillende instances lijkt op zich zeer aantrekkelijk, maar men moet er wel rekening mee houden dat dit vrij snel heel wat van de systeembronnen vergt. Terwijl een SQL Server instance initieel minder geheugen gebruikt dan een Oracle instance, met name omdat het dynamisch wordt beheerd, moet men er rekening mee houden dat de verschillende instances een invloed kunnen hebben op elkaars performance. Net zoals bij DB2 en in tegenstelling tot Oracle, is het mogelijk om in één instance meerdere databases te beheren. Voor de performance is het echter efficiënter om het aantal instances terug te brengen en de gegevens in verschillende databases onder te brengen.

Om zich met een instance te verbinden, gebruikt men de naam van de Windows server, aangevuld met de naam van de instance.

Architectuur

In figuur 1 schetsen we schematisch een SQL Server database-omgeving.

Figuur 1: Een SQL Server instance



Zoals u kan zien op het schema, bestaat de instance uit twee grote delen, de code area die de code bevat, de kernel, de stack, ... en de memory pool. Deze laatste bestaat uit standaardonderdelen eigen aan alle databases: buffer, log, procedure cache, ... Net zoals bij andere Microsoft producten, draait SQL Server zeer efficiënt zonder aanpassingen aan de basisinstellingen en is het dus niet nodig om SQL Server te "tunen" voor efficiëntieredenen; toch is het mogelijk om de nodige informatie te zien en om instellingen op zeer laag niveau aan te passen.

Databases: vier, vijf, zes ...

Tijdens het aanmaken van een instance, creëert het systeem een SQL Server vrij van gebruikersgegevens. Toch bevat deze al een aantal databases: de Master, Msdb, Tempdb, Model databases bestaan reeds. (zie figuur 1). Ze worden de systeem-databases genoemd.

Deze zijn nodig voor de werking van SQL Server en ze mogen niet worden verwijderd. Twee andere databases: Pubs en Northwind worden eveneens aangemaakt, maar deze mogen worden verwijderd. Dit zijn voorbeeld-databases.

Vervolgens kan de DBA de gebruikers-databases die hij wenst aanmaken binnen eenzelfde instance.

Elke database, of het nu een systeem-database of een gebruikers-database betreft, bestaat steeds uit twee elementen, de gegevens en de log, elk vertegenwoordigd door één of meerdere bestanden op de schijf. Het aanmaken van een database is heel eenvoudig en gaat zeer vlot, aangezien het volstaat om de model-database te kopiëren. De nieuwe database neemt dus de karakteristieken over van de modeldatabase. In tegenstelling tot DB2, heeft een database dus een fysieke implementatie van zodra ze wordt aangemaakt; maar het initiële schijfgebruik is zeer klein, beginnende bij slechts 2 MB, 1 voor de gegevens en 1 voor de log.

Zonder in detail te treden, kunnen we vaststellen dat sommige zaken op het niveau van de instance gebeuren, terwijl andere op het niveau van de database worden bijgehouden.

- De logging gebeurt op het niveau van de database, zoals bij DB2 UDB; elke database heeft dus minstens één eigen logbestand.

- De catalog staat deels op de Master database en deels op de andere databases, ongeacht of het om systeem- of gebruikers-databases gaat. De informatie eigen aan de database zelf zit in de systeemtabellen van de database, de informatie betreffende de SQL Server zit in de systeemtabellen van de Master database.

- De tijdelijke opslag gebeurt op niveau van het instance (cf. DB2 for z/OS), in een speciaal daarvoor vooziene database, TempDB.

- De informatie over jobs, alerts, operators en opslag, zitten in de Msdb-database.