ABIS NIEUWSBRIEF VOOR DB2 PROFESSIONALS

JAARGANG 1 - NUMMER 5, FEBRUARI 2003

OPEN CURSOR

Na een paar weken relatieve rust, zijn we er weer klaar voor: een nieuwe reeks van 'Exploring DB2'.
Zoals steeds wordt op het eind van een werkjaar, de interne projectplanning voor het volgende werkjaar vastgelegd. Binnen het DB2 domein zullen bij ABIS twee onderwerpen centraal staan.
Enerzijds willen we aandacht besteden aan integratie, met producten zoals DB2, Java, MQ en .Net.
Anderzijds zullen we minder gekende en gebruikte DB2 features uitdiepen: User Defined Types, Procedurele objecten en SQL/PL, LOBS, Object en OLAP extensies.
Om maar een paar onderwerpen op te sommen. Onderwerpen waarover we in 'Exploring DB2' verslag zullen doen.
U merkt het - weer veel om naar uit te kijken!

Het ABIS team.

IN DIT NUMMER:

- Hoe hints combineren met de mogelijkheden geboden door DSN_STATEMNT_TABLE in het kader van performantie analyse: *DSN_STATEMNT_TABLE deel 2*.
- *Dossier 7* - sorteren op expressies ! ?
- *Identities deel 2* - Toch niet zo evident?
- Het tweede artikel rond *Federated Databases* is verschoven naar volgende editie.
- *Cursusplanning mrt 2003 - mei 2003*.

CLOSE CURSOR

In het volgende nummer besteden we in detail aandacht aan een feature dat aan de basis ligt van een aantal nieuwe mogelijkheden in DB2: Large Objects (LOBs). En u mag ook het tweede deel van Text Extenders verwachten.

Tot dan!

DSN_STATEMNT_TABLE - 2

Eric Vennmans (ABIS)

Het gebruik van de DSN_STATEMNT_TABLE (DST) werd beschreven in een vorig artikel (zie Nr. 3 van november 2002). Er werd behandeld hoe de DB2 kostenschatting kan helpen:

- om de beste (goedkoopste, snelste, ...) oplossing te vinden, wanneer meerdere SQL varianten in aanmerking komen om DB2 de gewenste actie te laten uitvoeren,
- om verschillen in toegangspaden aan te tonen, wanneer de DB2 EXPLAIN functie dit niet doet.

In de inset vindt u ter opfrissing een compact overzicht van wat de DST te bieden heeft.

De DSN_STATEMNT_TABLE heeft als belangrijkste kolommen:

QUERYNO	INTEGER
COST_CATEGORY	CHAR(1)
PROCMS	INTEGER
PROCSU	INTEGER

Naast de identificatie van de query waarvoor kosteninformatie wordt gegeven, krijgen we:

- PROCMS, PROCSU: de geschatte kosten worden uitgedrukt in 'milliseconden' en in 'service units',
- COST_CATEGORY: 'A' duidt op een relatief nauwkeurige kostenberekening (alle nodige informatie is beschikbaar); 'B' duidt op een 'twijfelachtig' resultaat,
- REASON: geeft een uitleg die hoort bij dit 'twijfelachtige' resultaat (categorie 'B').

Beperkingen van de DB2 kostencalculatie

Het gebruik van de resultaten die DB2 in de DST publiceert, is niet altijd evident. Er zijn een aantal situaties waarbij een 'gewone' kostencalculatie een vertekend beeld geeft. Dit is het geval:

- bij het gebruik van variabelen in 'embedded static SQL',
- bij berekeningen op basis van 'misleidende' statistieken,
- bij berekeningen voor queries die maar gedeeltelijk worden uitgevoerd,
- bij queries (beperkt in aantal) waar de DB2 berekeningen fout zijn.

In dit artikel gaan we dieper in op de eerste situatie (static SQL). De andere komen later aan bod in een vervolgartikel.

Eenvoudig onderzoek i.v.m. static SQL

In programma's wordt doorgaans gewerkt met 'static SQL'. In deze vorm van SQL, wordt meestal gebruik gemaakt van voorwaarden ('where'-conditions) met variabelen op de plaats waar bij het uitvoeren 'values' thuishoren. De keuze van toegangspaden voor deze SQL, gebeurt normaal tijdens het 'BIND'-proces. Op dat ogenblik zijn de variabelen nog niet ingevuld. De 'values' zijn dus onbekend. Nochtans spelen ze voor heel wat queries een cruciale rol bij het bepalen van het toegangspad. De meeste 'filterfactoren' worden namelijk afgeleid van deze 'values' en het zijn deze 'filterfactoren' die in sterke mate de keuze van het toegangspad door de DB2 optimizer beïnvloeden.

Wanneer 'values' en bijhorende 'filterfactoren' een rol spelen bij het kiezen van een toegangspad, lost DB2 het ontbreken ervan op door gebruik te maken van 'default values'. Deze worden bepaald of berekend volgens algoritmes die gepubliceerd staan in de 'DB2 Administration guide: Performance monitoring and tuning '.

Als de 'default values' overeenkomen met de values die 'normaal' gebruikt worden wanneer het betrokken programma zal worden uitgevoerd, is de keuze van het toegangspad en de bijhorende kostencalculatie van DB2 betrouwbaar (tenzij één van de andere bovenvermelde situaties oorzaak is van een vertekend beeld).

Is er echter een verschil tussen wat DB2 veronderstelt en de verwachte, reële situatie, dan kan de bijhorende kostenberekening 'irrelevant' zijn. We geven een eenvoudig voorbeeld waarbij de kostencalculatie een vertekend beeld geeft (zie voorbeeld 1).

Voorbeeld 1

```
EXEC SQL
   DECLARE XSelect CURSOR FOR
        SELECT    *
            FROM     XTable
            WHERE XDatum > :DatumVariabele
END-EXEC

XDatum: COLCARD, FIRSTKEYCARDF, FULLKEYCARDF  == 1500

PLAN_TABLE en DST na explain:

QNo  Method  Table  Access  Match Index   Index Sort Sort Prefetch
             Name           Cols  Name    Only  N    C
100  0       XTABLE I       1     IXDATUM N     NNNN NNNN L

QNo    SqlType   Category MSec    SUnits      Reason
100    SELECT    B        6910    14246       HOST VARIABLES
```

In het toegangspad wordt het gebruik van een index (met 'List Prefetch') beschreven om de voorwaarde in de query te controleren. Dit is voor bovenstaande query meestal een aanvaardbaar toegangspad.

Enkel als de waarde in de DatumVariabele klein is, en de query dus veel rijen oplevert, komt het gebruik van de index ter discussie te staan.

Even veronderstellen dat in de gebruikte omgeving dit toegangspad de beste oplossing is. Rest nog de vraag of het goed genoeg is. Voor kritieke toepassingen wordt deze vraag best beantwoord vooraleer men aan het programmeren gaat. Misschien moet men in de applicatiecoding het minder snel uitvoeren van de query expliciet opvangen.

In het voorbeeld is zoals reeds gezegd, het toegangspad het best haalbare, maar de gepubliceerde kost is 6910 msec. Voor een online transactie is dit allicht te hoog.

De waarde voor de kostencalculatie is door DB2 echter berekend op basis van een default filterfactor (zie Reason = HOST VARIABLES). DB2 berekent op basis van de kolomcardinality (1500), een default filterfactor van 1/30 (zie hiervoor de reeds vermelde DB2 Administration Guide). M.a.w. DB2 veronderstelt dat deze query gemiddeld 3,33...% van de rijen oplevert als resultaat. De geschatte kosten voor het ophalen van deze hoeveelheid informatie, wordt in de DST gepubliceerd.

Wanneer we echter weten dat de toekomstige gebruikers van de applicatie waarin deze query zal worden gebruikt, enkel geïnteresseerd zijn in zeer recente informatie, zal de echte filterfactor veel kleiner zijn, dus ook de kost. Om een correcter beeld te krijgen van deze reële kost, kunnen we dezelfde query onderzoeken met expliciet een recente datum i.p.v. een variabele. Zie hiervoor voorbeeld 2.

Voorbeeld 2

```
Explain plan set queryno = 101 for
SELECT  *
    FROM     XTable
    WHERE XDatum > '10.01.2003'

QNo  Method  Table  Access  Match Index   Index Sort Sort Prefetch
             Name            Cols  Name    Only  N    C
101  0       XTABLE I        1     IXDATUM N     NNNN NNNN L

QNo    SqlType   Category MSec    SUnits      Reason
101    SELECT    A        172     353
```

We zien dat DB2 nog steeds hetzelfde toegangspad gebruikt, maar dat de berekende kost significant lager ligt. DB2 gebruikt nu een berekende filterfactor i.p.v. een default. Het verschil tussen '10.01.2003' en de op één na hoogste datum (in de Catalog te vinden als HIGH2KEY value voor de betrokken kolom) wordt vergeleken met de totale spreiding van de waarden in de betrokken kolom (verschil tussen LOW2KEY en HIGH2KEY voor de XDatum kolom). In ons voorbeeld wordt zonder verder de detailcijfers te geven, door DB2 berekend dat om en bij de 0,1% van de rijen in het resultaat van de query terechtkomen.

De hiervoor berekende kost (172 msec) is zeker aanvaardbaar in een online transactie. Dus de applicatie heeft voor deze query geen bijkomende tuning nodig. Een opervlakkige controle (zie opnieuw voorbeeld 1), suggereerde dit wel.

Het gegeven voorbeeld is echter erg triviaal. Iemand met enige 'feeling' voor DB2 kan ook zonder onderzoek met reële waarden, tot dezelfde conclusie komen. Hier gaat het echter over de manier van werken. Voor meer complexe queries kan het onderzoek met reële waarden wel nodig zijn om een betere kijk te krijgen op de haalbaarheid ervan.

Complexer onderzoek

Heel wat queries laten een eenvoudig onderzoek, zoals hierboven beschreven, niet zonder meer toe. Immers, bij het onderzoeken van een query kan het toegangspad veranderen door het gebruik van expliciete values (zie voorbeeld 3). Nu wordt de beoordeling een stuk moeilijker.

Voorbeeld 3

```
EXEC SQL
           DECLARE YSelect CURSOR FOR
                SELECT    *
                   FROM     YTable
                   WHERE YNaam LIKE :NaamVariabele
                          AND  YStatus = '0'
        END-EXEC

QNo  Method  Table  Access  Match Index   Index Sort Sort Prefetch
             Name           Cols  Name    Only  N    C
200  0       YTABLE M       0             N     NNNN NNNN L
200  0       YTABLE MX      1     IXSTAT  Y     NNNN NNNN S
200  0       YTABLE MX      1     IXNAAM  Y     NNNN NNNN
200  0       YTABLE MI      0             N     NNNN NNNN

QNo    SqlType   Category MSec    SUnits      Reason
200    SELECT    B        5568    11484       HOST VARIABLES

Verplicht gebruik van 5 letters in de naam:

Explain plan set queryno = 201 for
SELECT    *
    FROM     YTable
    WHERE YNaam LIKE 'VERBR%'
           AND  YStatus = '0'

QNo  Method  Table  Access  Match Index   Index Sort Sort Prefetch
             Name           Cols  Name    Only  N    C
201  0       YTABLE I       1     IXNAAM  N     NNNN NNNN

QNo    SqlType   Category MSec    SUnits      Reason
201    SELECT    A        4       9
```

De kostenindicatie voor query 201, hoe interessant ook, is niet relevant. In de applicatie zal het toegangspad van deze queryno immers niet gekozen worden. We zullen daar te maken hebben met het toegangspad van queryno = 200 omdat we moeten werken met variabelen.

Indien we toch een reële kostenindicatie willen voor dit toegangspad, moeten we met 'hints' werken (zie Nr.1 van september 2002). We gebruiken een 'hint' indicatie om het oorspronkelijke toegangspad vast te houden. Wanneer nu een kostencalculatie gevraagd wordt voor deze query met een expliciete value, zal de kostencalculatie weer wel relevant zijn voor wat in de applicatie zal gebeuren (zie voorbeeld 4).

Voorbeeld 4

```
EXEC SQL
           DECLARE XSelect CURSOR FOR
                SELECT    *
                   FROM     YTable
                   WHERE YNaam LIKE :NaamVariabele
                          AND  YStatus = '0'
        END-EXEC

QNo  Method  Table   Access  Match Index   Index Opthint Hint
             Name            Cols  Name    Only          used
200  0       YTABLE M       0             N     Q200
200  0       YTABLE MX      1     IXSTAT  Y     Q200
200  0       YTABLE MX      1     IXNAAM  Y     Q200
200  0       YTABLE MI      0             N     Q200

QNo    SqlType   Category MSec    SUnits      Reason
200    SELECT    B        5568    11484       HOST VARIABLES

SET CURRENT OPTIMIZATION HINT = 'Q200'

Explain plan set queryno = 200 for
SELECT    *
    FROM     YTable
    WHERE YNaam LIKE 'VERBR%'
           AND  YStatus = '0'

QNo  Method  Table   Access  Match Index   Index Opthint Hint
             Name            Cols  Name    Only          used
200  0       YTABLE M       0             N             Q200
200  0       YTABLE MX      1     IXSTAT  Y             Q200
200  0       YTABLE MX      1     IXNAAM  Y             Q200
200  0       YTABLE MI      0             N             Q200

QNo    SqlType   Category MSec    SUnits      Reason
200    SELECT    A        412     851
```

We kunnen opnieuw besluiten dat de reële kost (412 msec) van de geprogrammeerde SQL aanvaardbaar is voor online transacties.

Uit de resultaten van het onderzoek, dat in voorbeeld 4 wordt beschreven, blijkt de beste oplossing uiteindelijk deze met het aangepaste toegangspad (zie query 201). Deze oplossing wordt echter niet spontaan gebruikt door DB2 voor de geprogrammeerde 'static

SQL' versie. Willen we absoluut deze goedkoopste oplossing (er zijn complexere SQL queries waarbij dit wel degelijk gewenst wordt) dan kunnen we eveneens 'hints' gebruiken. Deze keer om een toegangspad 'op te leggen' voor de geprogrammeerde versie. Er is echter een groot verschil met het eerste gebruik, zoals beschreven in voorbeeld 4. Immers, voor het kostenonderzoek gebruiken we een 'hint' slechts één maal om de kostenindicatie te krijgen voor een bijgestuurd toegangspad. In het tweede geval (voor het goedkoopste toegangspad) moet de 'hint' permanent blijven functioneren. Dit laatste is niet eenvoudig. De 'hint'-aanduiding staat immers in de PLAN_Table, niet in het programma. Bij een eventuele ongewilde verwijdering ervan, zal DB2 bij de eerste de beste gelegenheid (d.w.z. een BIND of REBIND) terugvallen op het 'oude' toegangspad. Functioneel zal alles in orde blijven, maar de performance waar het hier om te doen is, zal allicht terugvallen beneden het gewenste niveau.

Besluit

De DST bevat interessante kosteninformatie om het tunen van kritische SQL queries te ondersteunen. Om een 'relevante' kostenindicatie te krijgen, moeten we voor een aantal queries bijkomend onderzoek verrichten. Echter, voor wie een goed zicht heeft op het EXPLAIN gebeuren en aanverwante zaken, is dit nauwelijks een probleem.

DOSSIER 7

Order by ...

Verwacht u ook een negatieve sqlcode bij het uitvoeren van volgend sql statement?

```
select sum(inkomsten), fk_cid
from sessions
group by fk_cid
order by sum(inkomsten)
```

Neen! Niet langer! Sorteren kan nu ook rechtstreeks op expressies. Enig idee waarom u het zou gebruiken?

Tom Avermaete (ABIS)

1,2,3,4,5....en wat daarna? Over identity kolommen - 2

Katrien Platteborze (ABIS)

In het eerste nummer van Exploring DB2 werd de definitie van de identity kolom besproken, een nieuwe feature in DB2 for OS/390 V7. In deze editie, gaan we verder in op deze identity kolom met het gebruik van de cache optie en het belang van de cataloog kolom maxassignedval in sysibm.syssequences, het alter table statement, en het unloaden/loaden van tabellen met een identity kolom. In de inset vindt u een korte beschrijving van het begrip identity.

Een identity kolom is een kolom waarvoor DB2 data genereert. De waarden die gegenereerd worden zijn afhankelijk van de opties die men instelt. Men moet een startwaarde kiezen, een waarde waarmee er verhoogd wordt, een maximum waarde en een minimum waarde. Men moet ook aanduiden of men bij uitputting terug vooraan bij de startwaarde mag beginnen en of er waarden in memory gecached worden. De kolom kan aangemaakt worden met generated always (DB2 genereert altijd een waarde) or generated by default (men kan zelf ook waarden inserten).

We vestigen nog eens de aandacht op de volgende punten:

- per tabel 1 identity kolom;
- datatype: smallint, int of decimal met scale 0;
- de kolom krijgt automatisch een not null definitie;
- identity dwingt geen uniekheid af, daarvoor is een unieke index nodig. Wanneer men met de 'cycle' optie werkt of zelf waarden toevoegt kunnen er dubbele waarden voorkomen;
- wanneer men zelf waarden toevoegt houdt DB2 hier geen rekening mee, DB2 baseert zich op de door hem laatst toegevoegde waarde;
- ook wanneer er rijen gedeleted worden houdt DB2 hier geen rekening mee;
- identity kolommen kunnen niet gewijzigd worden, hiervoor is een drop en create tabel nodig;

```
CREATE TABLE tableA
(PKeyColumn  smallint  NOT
NULL
GENERATED ALWAYS AS
IDENTITY
(START WITH 1,
 INCREMENT BY 1, CACHE 20,
 NO CYCLE, MINVALUE 0,
 MAXVALUE 32767)
 ....
```

Het syntax kader toont een voorbeeld van een tabel declaratie met identity kolom.

In wat volgt worden op basis van ervaringen en vaststellingen uit de praktijk, minder evidente karakteristieken van identities toegelicht.

Het 'insert' statement

```
insert into idloada
overriding user value
select * from idloadd;
```

Wanneer men een insert statement wil uitvoeren in een tabel met identity op basis van waarden die zich bevinden in een andere tabel met gelijkaardige structuur dan zal dit lukken wanneer de identity kolom 'generated by default' gedefinieerd is. Indien we werken met 'generated always' dan kan men het probleem oplossen door de optie 'overriding user value' toe te voegen aan het insert statement. In de syntax kader vindt u een voorbeeld. De tabel idloada heeft een identity kolom aangemaakt met de 'generated always' optie.

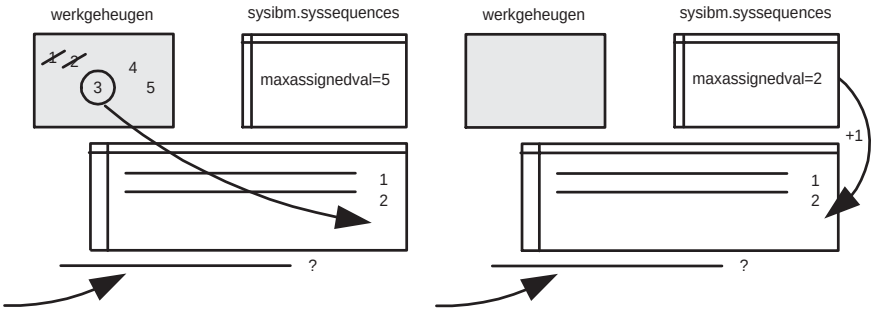
cache - maxassignedval

De cache optie specificeert het aantal waarden voor de identity kolom die DB2 voorbereidt in het geheugen - cached - waardoor ze door een applicatie sneller kunnen worden aangewend. De kleinste cache waarde die kan gespecificeerd worden is 2, de default waarde is 20. Caching kan worden gedeactiveerd - gebruik de 'no cache' optie.

In de kolom maxassignedval van de sysibm.syssequences catalog tabel duidt DB2 de maximum waarde aan die op dat ogenblik gepreallocceerd is.

Wanneer er niet gepreallocceerd wordt, duidt deze waarde het laatste gegenereerde getal aan, en zal DB2 op basis van deze waarde een nieuwe genereren bij het volgende insert statement. De volgende vragen dringen zich op: Wat is de invloed van transactie management en wat zijn mogelijke oorzaken van niet-sequentiële waarden?

Figuur: In het linker deel van de tekening wordt 3 toegekend als volgende waarde op basis van nog niet gebruikte waarden uit het werkgeheugen. In het rechter deel wordt er met no cache gewerkt en op basis van het getal in de maxassignedval kolom wordt de volgende waarde berekend.



Gegenereerde waarden zijn systeemgericht en niet transactie gerelateerd: rollback en commit hebben geen invloed op de toegekende waarden:

- Wanneer een rollback wordt uitgevoerd zullen één of meerdere waarden overgeslagen worden. In het geval van de 'no cache' optie wordt de maxassignedval niet mee gerollbacked. DB2 baseert zich op deze niet gerollbackedte waarde voor het genereren van de volgende identity waarde. Er is ook geen rollback van de gebruikte gepreallocceerde waarden wanneer de 'cache optie' gebruikt wordt.

- Het is het ogenblik van de insert die de waarde van het gegenereerde getal bepaalt en niet het ogenblik waarop de transactie waarvan ze deel uitmaakt, commit.

- Het feit dat de waarde in maxassignedval niet terug gedraaid wordt heeft ook een gevolg voor het backup & recovery mechanisme. Op tijd1 werden waarden tot 5 toegekend voor de indentity kolom. Daarna kent DB2 nog eens waarden toe tot 10. Als men nu recoveret naar tijd1 dan zullen na de recovery waarden vanaf 11 toegekend worden. De waarden tussen 5 en 11 ontbreken.

Er zijn nog andere oorzaken van niet-sequentiële waarden:

- Ook bij systeemproblemen kunnen waarden uit de cache verdwijnen zodanig dat deze nooit zullen gebruikt worden. In een datasharing omgeving kan een specifiek probleem optreden: de eerste DB2 kan waarden 1 tot 5 (cache 5) toegekend krijgen, de tweede DB2 waarden 6 tot 10. Het is afhankelijk van de volgorde van de transacties van de twee systemen in welke volgorde de waarden van de identity kolom zullen toegekend worden. Het resultaat zou iets dergelijks kunnen zijn: 1, 2, 6, 7, 8, 9, 10, 3, 4, 5. Dit betekent dat vanuit de eerste DB2 eerst twee insert transacties uitgevoerd worden, dan komen er vijf vanuit de tweede DB2 en dan is het weer aan de eerste DB2. De no cache optie biedt een oplossing voor dit probleem.

- Bij het stoppen van de database gaat de informatie in cache niet verloren, uiteraard wel bij het stoppen van DB2. DB2 baseert zich volledig op de waarde in de maxassignedval kolom om nieuwe waarden te generen.

Toevoegen van een kolom

Wanneer men een gewone kolom toevoegt aan een tabel die niet leeg is, worden de default waarden gebruikt - dit is uiteraard niet aan te raden - of indien er geen default gedefinieerd is, worden het null waarden. Een kolom toevoegen zonder default en met een 'not null' clause geeft een foutmelding. Wanneer men een identity kolom toevoegt, wordt de tablespace in REORG pending (REORP) gezet. Men kan nu geen queries meer uitvoeren en ook de meeste utilities kunnen niet meer uitgevoerd worden. De REORP toestand kan verwijderd worden door het reorg utility te draaien. Tijdens de reorg gaat DB2 waarden voor de identity kolom genereren en de maxassignedval bepalen. Uiteraard zal een lege tabel niet in deze pending toestand geplaatst worden.

Hou rekening met de volgende opmerkingen:

- als max value in de identity definitie kleiner is dan het aantal geïnserted rijen dan gaat de reorg in abend en komt men in een situatie terecht die niet gemakkelijk op te lossen is.

- tijdens deze reorg wil DB2 een image copy nemen; indien men geen syscopy kaart meegeeft weigert DB2 de reorg uit te voeren. DB2 gaat niet in copy pending zoals men zou verwachten naar analogie met het load utility.

unload - load

Wanneer men bij het laden van een tabel niet voor elke kolom een waarde voorziet zal DB2 een waarde toekennenn voor de identity kolom. Indien de betrokken kolom gedefinieerd is met generated always, dan is het zelfs onmogelijk om zelf een waarde toe te kennen.

Wanneer men met het unload utility load statements en input laat genereren, dan zijn die rechtstreeks bruikbaar voor de load. Dit geldt voor de twee types identity kolommen en is in tegenstelling met wat gedocumenteerd is in de DB2 manuals: Er wordt niet met de optie 'ignorefields' en een dummy kolom gewerkt. In de syntax kaders vindt men de gegenereerde load statements voor de tabellen idloada en idloadd. Bij de tabel met identity kolom generated always wordt de identity kolom niet vermeld, voor het andere type van identity kolom wordt ze wel vermeld. Men kan deze natuurlijk weglaten wanneer men wil dat DB2 waarden genereert. In beide gevallen bevatten de datafiles waarden voor de identity kolom maar wanneer de identity kolom niet vermeld wordt in het load statement worden deze data genegeerd.

Voorbeeld

```
create table idloada
(  col1 smallint generated always as identity  (start with 1),
   col2 char(2)) in database abis;

 LOAD DATA INDDN SYSREC   LOG NO  RESUME YES
  EBCDIC  CCSID(00500,00000,00000)
  INTO TABLE "TB00020 "."IDLOADA           "
  WHEN(00001:00002 = X'005A')
  ( "COL2              " POSITION(  00006:00007) CHAR(002)
                           NULLIF(00005)=X'FF'
  )

create table idloadd
(col1 smallint generated by default as identity (start with 1),
 col2 char(2)) in database abis;

 LOAD DATA INDDN SYSREC   LOG NO  RESUME YES
  EBCDIC  CCSID(00500,00000,00000)
  INTO TABLE "TB00020 "."IDLOADD           "
  WHEN(00001:00002 = X'005D')
  ( "COL1              " POSITION(  00003:00004) SMALLINT
  , "COL2              " POSITION(  00006:00007) CHAR(002)
                           NULLIF(00005)=X'FF'
  )
```

CURSUSPLANNING MRT - APR - MEI 2003

DB2 for OS/390, een totaaloverzicht	1625 EUR	24-28/03 (L), 07-11/04 (W), 22-28/05/2003 (W)
DB2 UDB, een totaaloverzicht	1625 EUR	24-28/03 (L), 22-23/05 & 02-04/06 (W)
RDBMS concepten	325 EUR	24/03 (L), 07/04 (W), 22/05 (W)
Basiskennis SQL	325 EUR	25/03 (L), 08/04 (W) 23/05 (W)
DB2 for OS/390 basiscursus	975 EUR	26-28/03 (L), 09-11/04 (W), 26-28/05 (W)
DB2 UDB basiscursus	975 EUR	26-28/03 (L))
DB2 UDB concepten	375 EUR	07/03 (W)
SQL workshop	700 EUR	24-25/02 (L), 22-23/04 (W),
DB2 for OS/390 programmering voor gevorderden	1050 EUR	02-04/04 (W), 19-21/05 (L)
DB2 for OS/390: SQL performance	1200 EUR	03-05/03 (W)
Fysiek ontwerp van relationele databases	700 EUR	28-29/04 (L)
DB2 for OS/390 database administratie	1600 EUR	18-21/03 (W), 12-15/05 (L)
DB2 UDB database administratie	1600 EUR	17-20/03 (W), 05-08/05 (L)
DB2 UDB systeembeheer en performance	400 EUR	21/03 (W), 09/05 (L)
DB2 for OS/390 V7 upgrade voor ontwikkelaars	375 EUR	28/02 (L), 06/03 (W)
DB2 UDB en zijn extenders: XML en text search	200 EUR	28/03 (W), 16/05 (L)
DB2 UDB integratie met MQSeries	200 EUR	28/03 (W), 16/05 (L)

Plaats: L = Leuven; W = Woerden
Details, andere data en bijkomende 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