ABIS NIEUWSBRIEF VOOR DB2 PROFESSIONALS

JAARGANG 1 - NUMMER 9, JUNI 2003

OPEN CURSOR

Dit is Exploring DB2 nr. 9 - het laatste nummer voor het zomerreces, en het laatste nummer van deze eerste jaargang.
De positieve reacties die wij nog steeds mogen ontvangen, zijn voor ons het bewijs dat de formule waaraan we dit eerste werkjaar hebben gewerkt, de goede is. Een publicatie voor elke DB2 specialist, met oog voor nieuwe features, applicatieontwikkeling, systeembeheer, performance en organisatie.
Wij nemen nu twee maanden vakantie, maar zitten toch niet helemaal stil: we willen ook werken aan 'vers' materiaal; om u op basis van concrete testen en analyses opnieuw een aantal op praktijk gestoelde artikels te kunnen aanbieden.

Prettige zomermaanden - tot in september!
Het ABIS team.

IN DIT NUMMER:

CLOSE CURSOR

In het volgende nummer voor het eerst aandacht voor DB2 versie 8 - vanaf dan regelmatig in *Dossier 8.* En we starten met een mini-reeks die iedereen zal boeien: verkeerde optimalisaties ...

Tot dan!

Large Objects in DB2 for OS/390 - 2

Diane Hendrix(ABIS)

In het eerste artikel over LOB's (zie Exploring DB2 nr. 6) bespraken we reeds enkele algemeenheden. Een opfrissing:

- LOB's (Large Objects) kunnen gebruikt worden om grote dataobjecten, zoals geluidsbestanden, beeldfragmenten, foto's of grote tekstbestanden op te slaan. De maximale grootte van een LOB-kolomwaarde is 2 GB.

- er bestaan 3 verschillende LOB-datatypes: CLOB (character large object), BLOB (binary large object) en DBCLOB (double byte character large object).

- de LOB-data worden niet bewaard in de eigenlijke tabel (basistabel), maar in een hulptabel (in een eigen LOB-tablespace). Vanuit de basistabel wordt via een LOB-descriptor verwezen LOB-waarde.

- LOB-kolommen kunnen benaderd worden via de gewone SQL statements (SELECT, INSERT, UPDATE, DELETE). Een DELETE van een rij met één of meer LOB-kolommen, resulteert in een deallocatie van de LOB-datapages. Een UPDATE van een LOB-waarde wordt door DB2 vertaald in een DELETE-actie, gevolgd door een INSERT-actie. Hierbij worden de LOB-datapages eerst gedealloceerd en vervolgens wordt de nieuwe LOB-waarde op dezelfde datapages weggeschreven.

Wat nu volgt is een meer technische bespreking:

- wat zijn LOB-locators en hoe worden ze gebruikt?
- hoe zit het met LOB's en locking?
- DB2 utilities en LOB's?

Voorbeelden? Zie *COBOL en LOB's* op p. 13

Hoe worden LOB's en LOB-locators gebruikt?

Als je vanuit een applicatie een LOB-waarde wil benaderen kan je op twee manieren te werk gaan (Figuur 1).

```
Declaratie in COBOL:
01 LB USAGE IS SQL TYPE IS
CLOB (100K).

Omzetting door precompiler:
01 LB.
02 LB-LENGTH PIC S9(9) COMP.
02 LB-DATA.
   49  FILLER PIC X(32767).
   49  FILLER PIC X(32767).
   49  FILLER PIC X(32767).
   49  FILLER PIC X(4099).
```

- Je brengt de volledige LOB-waarde binnen in een host variabele. Die moet dan wel groot genoeg zijn om de LOB-waarde te bevatten. In een IBM COBOL for OS/390 omgeving is de maximale grootte van een variabele echter beperkt tot 32767 bytes. Voor grotere LOB's zullen er dus meerdere variabelen moeten gedeclareerd worden (dit wordt gedaan door de precompiler).

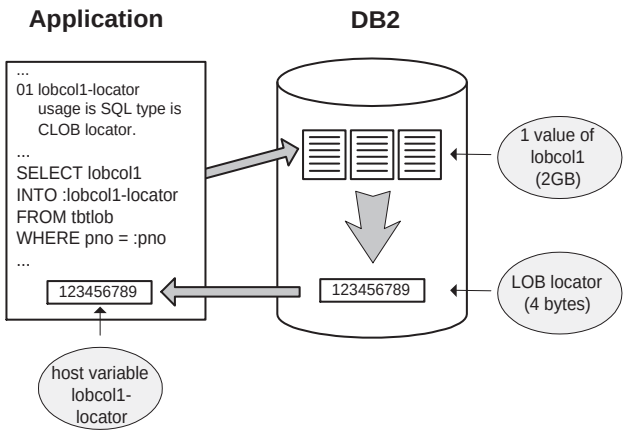
Wanneer men op deze manier te werk gaat, brengt DB2 de volledige LOB-waarde via de bufferpool binnen in de address space van de gebruiker. Als het over een grote LOB gaat neemt dit niet alleen veel ruimte in beslag, maar zal ook de performance verminderen!

```
Declaratie in COBOL:
01 LB-LOCAT USAGE IS SQL
TYPE IS CLOB-LOCATOR.

Omzetting door precompiler:
01 LB-LOCAT PIC S9(9) COMP.
```

▪Je gebruikt een LOB-locator. Door gebruik van een LOB-locator wordt slechts een 4-byte variabele gedefinieerd als een soort van interne referentie naar de echte LOB-waarde. In de applicatie kan perfect met de LOB-locator gewerkt worden overal waar men ook met een host variabele kan werken.

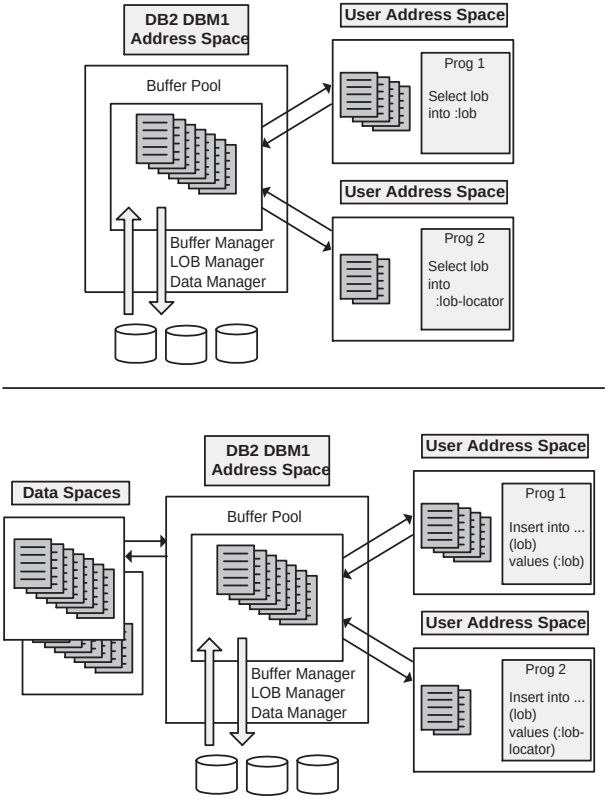
Figuur 1: gebruik van een LOB-locator



Afhankelijk van wat je in je applicatie met de LOB-locator doet, zal DB2 de LOB-waarde moeten materialiseren. Met materialisatie bedoelen we dat DB2 de LOB-waarde toch in het geheugen laadt in een zogenaamde dataspace. Een dataspace is een speciaal door DB2 gealloceerde ruimte om LOB-waarden te materialiseren. De grootte van de dataspace zal afhangen van het aantal LOB-waarden dat moet gematerialiseerd worden en de grootte van de LOB-waarde(n). Over het algemeen probeert DB2 de materialisatie zolang mogelijk uit te stellen.

Zoals te zien is in Figuur 2 gebeurt er geen materialisatie bij een SELECT en DELETE, maar wel bij een INSERT en UPDATE.

Figuur 2: materialisatie van LOB-waarden in dataspaces



Welke voordelen biedt het gebruik van LOB-locators?

Door het gebruik van LOB-locators kan vermeden worden dat de address space van de gebruiker overladen wordt door het binnenbrengen van grote LOB-data. Enkel een kleine ruimte in de address space van de gebruiker zal ingenomen worden door de LOB-locator. Indien men in een distributed omgeving werkt, met de applicatie op de client en de data op de server, wordt aldus overbodig netwerkverkeer tussen server en client vermeden. Indien de LOB-data concreet getransporteerd zou moeten worden van server naar client zou dit leiden tot een enorme stijging in het resource- en tijdsverbruik. Maar ook wanneer de data op de host blijft, kan het gebruik van LOB-locators een stijging van de performance tot gevolg hebben. Denk maar even aan applicaties die slechts een deel van de LOB nodig hebben, of waar de LOB-waarden tijdelijk gebruikt worden. Of aan applicaties waarin erg grote LOB's gemanipuleerd worden. In sommige omstandigheden is het gebruik van LOB-locators zelfs noodzakelijk. Dit is

met name het geval wanneer de grootte van de volledige LOB-waarde de maximale grootte van een host variabele overschrijdt. In een IBM COBOL voor OS/390 omgeving is dit voor een LOB met een grootte van > 16 MB.

Hoe zit het met locking?

Met de introductie van LOB's in DB2 werden er ook twee nieuwe locktypes gedefinieerd: het S-LOB lock en het X-LOB lock. LOB-locking is in feite een combinatie van locks op de LOB-tablespace, het geassocieerde ROWID in de basistabel, het LOB-versienummer in de basistabel en een S- of X-LOB lock van de LOB-datapages. De S- en X-LOB locks zijn nodig om ervoor te zorgen dat er op alle momenten een consistentie is tussen de LOB-waarde en de geassocieerde rij in de basistabel.

Bij een gewone leesoperatie zullen de LOB-datapages in een S-LOB lock geplaatst worden. Dit lock wordt door DB2 behouden tot de volledige LOB-waarde opgehaald is. Een tweede applicatie kan echter gelijktijdig dezelfde LOB-waarde wijzigen of verwijderen. Dit is mogelijk doordat een DELETE in feite enkel een deallocatie van de LOB-datapages betekent. Het is pas wanneer alle S-LOB locks wegvallen, dat de fysieke verwijdering van de LOB-waarde mogelijk wordt. Bij een DELETE-actie wordt er ook een S-LOB lock genomen op de LOB-datapages. Dit is nodig in geval van een rollback-situatie. Het S-LOB lock zorgt ervoor dat de LOB-datapages in dat geval gewoon opnieuw gealloceerd kunnen worden. Zonder een S-LOB lock zou de gedealloceerde ruimte immers ondertussen gebruikt kunnen worden door een INSERT-actie!

Het X-LOB lock wordt genomen bij een INSERT of UPDATE-actie. Daarmee wordt voorkomen dat applicaties trachten de LOB-waarde te lezen alvorens de INSERT of UPDATE-actie vervolledigd is. Aangezien een UPDATE een combinatie is van een INSERT en DELETE-actie, wordt er hier eerst een S-LOB lock op de LOB-datapages genomen en vervolgens een X-LOB lock.

LOB's en de DB2 catalog

Vanaf DB2 V6 zijn er een aantal nieuwe catalog tabellen bijgekomen en zijn er aan enkele bestaande tabellen nieuwe kolommen toegevoegd.

- SYSIBM.SYSAUXRELS geeft een beschrijving van alle relaties tussen basistabellen en verwante hulptabellen.

- SYSIBM.SYSLOBSTATS geeft statistische informatie omtrent LOB-tablespaces (wordt opgevuld of gewijzigd na uitvoering van het RUNSTATS-utility op de LOB-tablespaces). De kolom ORGRATIO geeft een indicatie van de organisatiegraad van de LOB-tablespace.

- SYSIBM.SYSCOLUMNS geeft in de kolom COLTYPE het LOB-type (BLOB, CLOB of DBCLOB), in de kolom LENGTH vindt men altijd 4 (LOB-indicator) en in de kolom LENGTH2 wordt de maximale lengte van de LOB-kolom gegeven (zoals tijdens de creatie werd meegegeven).

▪ SYSIBM.SYSTABLES bevat één rij voor elke basistabel en één voor elke hulptabel. In de kolom RECLENGTH wordt voor de basistabel de huidige recordlengte gegeven (4 bytes voor de LOB indicator); voor de hulptabel is die waarde altijd 0. In de kolom TABLESTATUS bevindt zich de reden van een I (incomplete) in de STATUS kolom, met name: L = geen hulptabel of index op de hulptabel, P = geen parent index.

Vanaf DB2 UDB for OS/390 V7 worden door DB2 ook LOB's gebruikt in de catalog tabellen.

LOB's en utilities

Bij de introductie van LOB's in DB2 werden een aantal nieuwe utilities (CHECK LOB, REORG LOB) geïntroduceerd; en aan bestaande utilities zijn opties toegevoegd, specifiek voor het behandelen van LOB's (CHECK DATA, RUNSTATS, LOAD, UNLOAD).

De LOAD en UNLOAD(V7) utilities kunnen gebruikt worden om LOB's <= 32 KB te manipuleren (maximale recordlengte van een dataset in een OS/390 omgeving is 32767 bytes!). Hou er rekening mee dat er enkele bytes gebruikt worden voor lengte- en NULL-indicatoren!

▪ UNLOAD: laat toe om 1 of meerdere kolommen weg te schrijven in een host dataset. Terzelfdertijd wordt er ook een controlecard voor het LOAD utility aangemaakt. Tijdens de UNLOAD van een volledige tabel wordt de ROWID kolom ook mee uit de tabel geladen, maar in de controlecard wordt geen ROWID kolom voorzien (tenminste als de kolom gedefinieerd werd als GENERATED ALWAYS). Bij het laden worden nieuwe ROWID-waarden gegenereerd.

▪ LOAD: telkens er een rij in de basistabel wordt geladen, gebeurt er een insert in de hulptabel en wordt er een key toegevoegd aan de index op de hulptabel.

Het RUNSTATS utility zorgt ervoor dat de statische informatie i.v.m. LOB's aangepast wordt in o.m. de SYSIBM.SYSTABLES, SYSIBM.SYSCOLUMNS, en SYSIBM.SYSLOBSTATS tabellen. In deze laatste tabel is vooral de ORGRATIO kolom belangrijk. Een waarde van 1 is optimaal en duidt op een perfecte LOB organisatie; een waarde groter dan 1 duidt aan dat het REORG utility moet worden uitgevoerd. Dit is voornamelijk het geval na het opladen van een grote hoeveelheid grote LOB's in de tabel. RUNSTATS moet ook worden gedraaid op de LOB tablespace!

Het REORG LOB utility zal ervoor zorgen dat de niet gebruikte vrije ruimte verwijderd wordt en dat de LOB-datapages terug aaneensluitend worden. De ORGRATIO kolom van de SYSIBM.SYSTABLES zal hierdoor terug op de optimale waarde 1 komen. Wel even opletten: tijdens een REORG van de basis tablespace wordt er niet aan de LOB's geraakt!

Het uitgebreide CHECK DATA utility en het nieuwe CHECK LOB utility kunnen uitgevoerd worden om :

- inconsistenties tussen de basis tablespace en de LOB-tablespace op te sporen. Enkele voorbeelden: *orphan LOB's* (naar een LOB-waarde in de LOB-tablespace wordt niet gerefereerd vanuit de basis tablespace), *ontbrekende LOB's* (een referentie in de basis tablespace verwijst naar een niet bestaande LOB-waarde in de LOB-tablespace), *out-of-sync LOB's* (naar een LOB-waarde in de LOB-tablespace wordt vanuit de basis tablespace gerefereerd, maar de LOB-indicator in de basistabel is NULL, of geeft lengte 0 weer), *ongeldige LOB's* (zoals vastgesteld na uitvoering van CHECK DATA met AUXERROR/INVALIDATE);

- om structurele defecten in een LOB-tablespace op te sporen of om ongeldige LOB-waarden terug te vinden.

Het CHECK LOB utility wordt uitgevoerd op de LOB-tablespace. Bij problemen zal de LOB-tablespace in een check pending (CHKP) of auxiliary warning (AUXW) status gebracht worden. De probleem LOB-waarden kunnen met een UPDATE of DELETE statement gecorrigeerd worden (i.g.v. AUXW). Of het REPAIR utility kan worden uitgevoerd om ongeldige LOB's te verwijderen (i.g.v. AUXW of CHKP). Heruitvoering van het CHECK LOB utility kan de CHKP status opheffen.

Het CHECK DATA utility wordt uitgevoerd op de basis tablespace en beschikt sinds DB2 V6 over de AUXERROR clausule. Deze clausule heeft voornamelijk een effect op wat er in geval van een ongeldige LOB-waarde, met die LOB gebeurt:

- AUXERROR(REPORT) (default): de basis tabel space wordt in een auxiliary check pending (ACHKP) status geplaatst. De basis tablespace wordt hierdoor onbruikbaar, maar de LOB-waarden kunnen gecorrigeerd worden met een SQL UPDATE of DELETE. Als alternatief is er het REPAIR utility. Heruitvoering van het CHECK DATA utility kan de ACHKP status opheffen.

- AUXERROR(INVALIDATE): de probleem LOB-kolom in de basistabel wordt als ongeldig gemerkt en de basis tablespace wordt in een AUXW status geplaatst. Om deze situatie te corrigeren: verwijder of wijzig de probleem LOB's met een SQL UPDATE of DELETE en voer het CHECK LOB utility uit. De AUXW status kan opgeheven worden door het CHECK DATA utility opnieuw uit te voeren.

Besluit

In het eerste artikel over LOB's werd reeds vermeld, dat LOB's goed doordacht gebruikt moeten worden. Zo moet men bijvoorbeeld opletten voor een daling van de performance. In dit tweede artikel hebben we gezien dat het werken met LOB-locators in plaats van met de eigenlijke LOB-variabelen zeker overwogen kan worden. In verband met performance, moet men ook opletten voor applicaties die LOB-inserts doen. Regelmatige uitvoering van het REORG LOB utility, gevolgd door een RUNSTATS is zeker noodzakelijk.

DOSSIER 7

Etcetera

Tot slot, een aantal weetjes!

Aan een SELECT-statement kan 'FETCH FIRST n ROWS ONLY' toegevoegd worden opdat DB2 niet de hele resultaatsverzameling zou ophalen. Een aantal queries kunnen hierdoor veel gemakkelijker geschreven worden.

```
select date from sessions
order by incomes
fetch first 2 rows only;
```

Wanneer men met een cursor positioned updates wil uitvoeren, dan kan men de 'FOR UPDATE'-clause op het SELECT-statement vermelden zonder kolomlijst. Dit geeft hetzelfde resultaat als wanneer men alle kolommen zou vermelden.

MAX en MIN kunnen ook gebruikt worden als scalaire functie:

```
select  MAX(incomes1, incomes2), incomes1, incomes2
from sessions;
```

Naast COUNT(*) en COUNT (distinct expression) kan de COUNT-functie ook gebruikt worden met alleen de kolomnaam: COUNT(expression) of COUNT(ALL expression). Het resultaat vermeldt het aantal niet NULL-waarden, inclusief de dubbele waarden.

UPDATE- en DELETE-statements kunnen naar zichzelf verwijzende subqueries bevatten.

DB2 kan join-technieken gebruiken voor subqueries in UPDATE- of DELETE-statements. Hierdoor kan de performance verbeterd worden. Er zijn wel een aantal eisen waar strikt aan voldaan moet zijn, het moet o.a. een gecorreleerde subquery zijn.

```
update clients a set ctel = null
where a.c_cono in (select cono
                   from companies b
                   where a.ctel = b.cotel);
```

DB2 is nu in staat te bepalen op welke kolommen sorteren zinvol is. In volgende query gaat hij niet op pa_cono sorteren - in V6 gebeurde dit nog wel.

```
select plname,pfname,pa_cono from persons
where pa_cono=3
order by pa_cono;
```

De MAX-kolomfunctie kan nu via de index direct de juiste waarde ophalen zonder deze helemaal te moeten doorlopen.

Men kan met scrollable cursors werken. Dit zijn cursors waarin men om het even welk record in de recordset kan ophalen. Indien men werkt met een resultaatstabel in een tijdelijke database kan men update en delete holes detecteren. Het voorkomen van een hole betekent dat de tabel waarop men fetch niet meer overeenstemt met de echte tabel.

In een stored procedure kan een COMMIT- of ROLLBACK-statement gebruikt worden.

Katrien Platteborze (ABIS)

Federated Databases - 2

Kris Van Thillo (ABIS)

In een vorig artikel (zie Exploring DB2 nr. 4) werden de basiselementen die aan de grondslag liggen van federated databases uiteengezet. Een aantal belangrijke componenten van de federated databaseserver-architectuur werden besproken: servers, wrappers, nicknames. In dit kort artikel staan we even stil bij een aantal factoren die een invloed hebben op applicatie-performance in een federated database.

Het gaat nog steeds om SQL...

Uiteindelijk moet nog steeds SQL worden uitgevoerd - door de federated database én door de (remote) datasource.

Een eerste aandachtspunt is dan ook de optimizatie van het SQL-statement door de federated database cost-optimizer. Hiertoe heeft deze nood aan metadata die van de datasource-cataloog wordt opgevraagd op het moment dat een nickname wordt aangemaakt. Specifiek vraagt de federated database volgende informatie op (deze wordt opgeslagen in de federated database-cataloog):

- beschrijving van in de datasource aanwezige indexen. Indien deze informatie niet kan worden bekomen, of indien deze info niet beschikbaar is (de nickname kan geen objecttranslaties uitvoeren op de datasource), of indien men de optimizer wil sturen, kan men dummy indexspecificaties op nicknames toevoegen aan de federated database. Dummy in de zin dat deze indexen niet worden gecreëerd op de federated server; ze worden enkel geregistreerd (i.e. create for specification only)!
- statistische informatie. Indien deze niet aanwezig is, of om deze up-to-date te houden met de steeds wijzigende inhoud van de datasource, kan men de federated database-cataloog met de bekende update-statements aanpassen! Deze statistische informatie is cruciaal voor de optimizer - op basis hiervan worden o.a. schattingen gemaakt m.b.t. het aantal rijen dat de datasource over het netwerk zal moeten doorzenden.

Vervolgens moeten we stilstaan bij het SQL-statement dat door de federated database naar de remote datasource wordt gestuurd, alsook bij de omvang van de data die door deze laatste wordt teruggestuurd. We kunnen het explain-utility in een of andere vorm gebruiken om een antwoord te krijgen op beide vragen. Belangrijke aandachtspunten in deze context zijn bijvoorbeeld: is de SQL die wordt verstuurd naar de datasource, optimaal voor de datasource? Welke elementen in de verwerking worden door de federated server zelf uitgevoerd, en waarom? En hoe worden eventuele resultaten die door verschillende datasources worden opgestuurd, gecombineerd - via bijvoorbeeld tijdelijke tabellen?

Andere factoren

Natuurlijk moeten we ook aandacht schenken aan andere factoren die een belangrijke rol spelen. Bijvoorbeeld:

- representatieve hardware: netwerk bandbreedte, capaciteit van de netwerk adapters, aantal CPU's, memory, ... spelen een belangrijke rol. Merk op dat naast de traditionele behoeften van de datasources, ook de federated database over voldoende resources moet beschikken - deze laatste is immers vaak verantwoordelijk voor het uitvoeren van een reeks taken die niet aan de datasource kunnen worden toevertrouwd, zoals extra sorteerwerk (al dan niet via tijdelijke tabellen), etc.

- de configuratie en tuning van de federated database - data caching (bufferpools), sorteerruimte in memory, en data blocking zijn in deze context belangrijke elementen.

- de configuratie en tuning van de datasource.

In 'CASE' u het zelf niet had gevonden ...

Vorige maand vroegen we u een oplossing te verzinnen voor het volgende vraagstuk: Selecteer de maanden die een aandeel van meer dan 20% in de omzet hebben. Velen onder u hebben ongetwijfeld het hoofd gebroken over deze vraag, slechts één SQL-schrijvend talent mailde een juiste oplossing naar de redactie: **Hielke Bijsma, ING (NL)** bedacht deze elegante oplossing:

```
SELECT CASE MAANDNR
              WHEN 1 THEN 'JANUARI'
              WHEN 2 THEN 'FEBRUARI'
              WHEN 3 THEN 'MAART'
              ...
              WHEN 12 THEN 'DECEMBER'
       END AS MAAND,
       MAANDTOTAAL * 100.00 / JAARTOTAAL AS OMZET
FROM (SELECT MAANDNR, MAANDTOTAAL, JAARTOTAAL
      FROM  (SELECT MAANDNR, SUM(DAGTOTAAL) AS MAANDTOTAAL
             FROM (SELECT MONTH(DATUM) AS MAANDNR,
                          TOTAAL_DAGINKOMSTEN AS DAGTOTAAL
                   FROM AUDIT_TABEL
                   WHERE YEAR(DATUM) = 2003) DAGTOTALEN
             GROUP BY MAANDNR) MAANDTOTALEN,
            (SELECT SUM(TOTAAL_DAGINKOMSTEN) AS JAARTOTAAL
             FROM AUDIT_TABEL
             WHERE YEAR(DATUM) = 2003) JAARTOTALEN
      ) OMZET
WHERE MAANDTOTAAL * 100.00 / JAARTOTAAL > 20
```

OVERZICHT VAN DE GEPUBLICEERDE ARTIKELS

CURSUSPLANNING SEP - DEC 2003

DB2 for OS/390, een totaaloverzicht	1625 EUR	15-19/09 (L), 13-17/10(W), 03-07/11(L), 08-12/12 (W)
DB2 UDB, een totaaloverzicht	1625 EUR	13-14&20-22/10 (W), 03-04&12-14/11 (L)
RDBMS concepten	325 EUR	15/09 (L), 13/10 (W), 03/11 (L), 08/12 (W)
Basiskennis SQL	325 EUR	16/09 (L), 14/10 (W), 04/11(L)
DB2 for OS/390 basiscursus	975 EUR	17-19/09 (L), 15-17/10 (W), 05-07/11(L), 10-12/12 (W)
DB2 UDB basiscursus	975 EUR	20-22/10 (W), 12-14/11(L)
DB2 concepten	375 EUR	24/10 (L), 09/12 (W)
DB2 UDB applicatieperformance	400 EUR	10/11 (W)
SQL workshop	700 EUR	29-30/09 (L), 27-28/10 (W), 17-18/11(L), 18-19/12 (W)
DB2 for OS/390 programmering voor gevorderden	1050 EUR	27-29/10 (W), 03-05/12 (L)
DB2 for OS/390: SQL performance	1200 EUR	03-05/11(W), 15-17/12 (L)
Database applicatieprogrammering met JAVA	800 EUR	20-21/10 (L), 13-14/11(W)
Tunen van JAVA - DB2 applicaties	800 EUR	27-28/11(W), 18-19/12 (L)
Fysiek ontwerp van relationele databases	700 EUR	06-07/11(L)
DB2 for OS/390 DBA	1600 EUR	20-23/10 (W), 02-05/12 (L)
DB2 UDB database administratie	1600 EUR	27-30/10 (W), 24-27/11(L)
DB2 UDB systeembeheer en performance	400 EUR	11/11 (W), 15/12 (L)
DB2 for OS/390 V7 upgrade voor ontwikkelaars	375 EUR	26/09 (L), 12/11(W)
DB2 UDB extenders: XML en Text	200 EUR	23/10 (W), 16/12 (L)
DB2 UDB integratie met MQSeries	200 EUR	23/10 (W), 16/12 (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

Bijlage

COBOL en LOB's

- Example 1 - INSERT van een tekstfile in een CLOB-kolom (gebruik van host variabelen);
- Example 2 - INSERT van een tekstfile in een CLOB-kolom (gebruik van een locator);
- Example 3 - SELECT van CLOB-waarde in een host variabele of een LOB-locator;
- Example 4 - INSERT van een CLOB-waarde die in de applicatie wordt samengesteld uit twee LOB-waarden;
- Example 5 - INSERT van een CLOB-waarde die een onderdeel is van een bestaande LOB-waarde;
- Example 6 - verwijdering van een specifiek onderdeel van een CLOB-waarde;
- Example 7 - toevoeging van een stuk tekst aan een bestaande CLOB-waard;
- Example 8 - extractie van een CLOB-waarde in een host file via een host variabele;
- Example 9 - extractie van een CLOB-waarde in een host file via een LOB-locator.

```
**************
* EXAMPLE  1 *
**************

       IDENTIFICATION DIVISION.
      ********************************************************
      * PROGRAMMMA OM EEN INSERT TE DOEN VAN EEN RIJ IN DE
      * TBTLOB TABEL
      * DEZE TABEL BEVAT TWEE LOB KOLOMMEN
      *     - LOBCOL1 VAN HET TYPE CLOB(100K)
      *     - LOBCOL2 VAN HET TYPE BLOB(200K)
      * DEZE TABEL BEVAT OOK EEN ROWID KOLOM (VERPLICHT EN GENERATED
      *                                       ALWAYS)
      * DE INPUT VOOR DE CLOB KOLOM IS EEN HOST INPUT FILE
      * (GEWONE TEKST FILE)
      * DE COMPLETE FILE WORDT VIA EEN LEESLUS IN DE HOSTVARIABELE
      * LOBCOL1-DATA  GEBRACHT EN DE LENGTE VAN HET VELD
      * LOBCOL1-LENGTH WORDT TELKENS AANGEPAST
      * NA DEZE LEESLUS WORDT EEN INSERT GEDAAN IN DE TABEL
      ********************************************************
       PROGRAM-ID.         LOBEX1.
       ENVIRONMENT DIVISION.
       INPUT-OUTPUT SECTION.
      *--------------------*
       FILE-CONTROL.
      *------------*
           SELECT LOB-FILE-IN  ASSIGN TO FILEIN.

       DATA DIVISION.
```

```
 FILE SECTION.
*------------*
 FD  LOB-FILE-IN.
 01  LOB-RECORD-IN       PIC X(80).

 WORKING-STORAGE SECTION.
* HOST VARIABLES
 01 SW-EOF           PIC X  VALUE 'N'.
    88 EOF                  VALUE 'Y'.
 01  INDICATOR PIC S9(4) COMP.
 01  RET-CODE  PIC S999 SIGN LEADING SEPARATE.
 01  PNO-IN         PIC 999.

* SQL DECLARES
     EXEC SQL
       INCLUDE SQLCA
     END-EXEC.

     EXEC SQL DECLARE TBTLOB TABLE
     ( PNO                          SMALLINT NOT NULL,
       LOBCOL1                      CLOB(102400),
       TBTLOBROWID                   ROWID NOT NULL,
       LOBCOL2                      BLOB(204800)
     ) END-EXEC.

 01  DCLTBTLOB.
     10 PNO                 PIC S9(4) USAGE COMP.
     10 LOBCOL1             USAGE SQL TYPE IS CLOB (100K).
     10 TBTLOBROWID         USAGE SQL TYPE IS ROWID.
     10 LOBCOL2             USAGE SQL TYPE IS BLOB-LOCATOR.


 PROCEDURE DIVISION.
 PROG-START.
*INLEZEN VAN FILE IN DE CLOB VARIABELE
     OPEN INPUT LOB-FILE-IN
     MOVE 0 TO LOBCOL1-LENGTH
     READ LOB-FILE-IN
        AT END SET EOF TO TRUE
     END-READ
     PERFORM UNTIL EOF
        MOVE LOB-RECORD-IN
            TO LOBCOL1-DATA(LOBCOL1-LENGTH + 1 : 80)
        ADD 80 TO LOBCOL1-LENGTH
        READ LOB-FILE-IN
          AT END SET EOF TO TRUE
        END-READ
     END-PERFORM
     CLOSE LOB-FILE-IN

* OPVULLEN VAN DE REST VAN DE KOLOMMEN
     ACCEPT PNO-IN
     MOVE PNO-IN TO PNO

*  INSERTEN VAN  DE CLOB-VARIABELE IN DE CLOB KOLOM
     EXEC SQL
        INSERT INTO TBTLOB
        (PNO,
         LOBCOL1)
      VALUES
        (:PNO,
         :LOBCOL1)
       END-EXEC
```

```
           IF SQLCODE NOT = 0
              MOVE SQLCODE TO RET-CODE
              DISPLAY 'FOUT BIJ INSERT'
              DISPLAY 'DE SQLCODE = ' RET-CODE
           END-IF

           STOP RUN.

**************
* EXAMPLE  2 *
**************

            IDENTIFICATION DIVISION.
      ***********************************************************
      * PROGRAMMMA OM EEN INSERT TE DOEN VAN EEN RIJ IN DE
      * TBTLOB TABEL
      * DEZE TABEL BEVAT TWEE LOB KOLOMMEN
      *     - LOBCOL1 VAN HET TYPE CLOB(100K)
      *     - LOBCOL2 VAN HET TYPE BLOB(200K)
      * DEZE TABEL BEVAT OOK EEN ROWID KOLOM (VERPLICHT EN GENERATED
      *                                       ALWAYS)
      * DE INPUT VOOR DE CLOB KOLOM IS EEN HOST INPUT FILE
      * ---> GEWONE TEKST FILE
      * WE VERONDERSTELLEN BIJ DEZE OEFENING DAT IN DEZE OMGEVING
      * SLECHTS EEN VARIABELE VAN 800 BYTES MOGELIJK IS
      * DAAROM WORDT IN DE LEESLUS EEN BIJKOMENDE LUS GECREEERD
     * WAARBIJ OM DE 10 RECORDS DE HOSTVARIABELE NAAR EEN LOCATOR WORDT
      * WEGGESCHREVEN.
      * OM DE 10 RECORDS WORDT ER EEN APPEND NAAR DE LOCATOR GEDAAN
      * HELEMAAL OP HET EINDE WORDT DE LOCATOR TOEGEVOEGD AAN DE
      * DATABASE
      ***********************************************************
       PROGRAM-ID.         LOBEX2.
       ENVIRONMENT DIVISION.
       INPUT-OUTPUT SECTION.
      *--------------------*
       FILE-CONTROL.
      *------------*
           SELECT LOB-FILE-IN  ASSIGN TO FILEIN.

       DATA DIVISION.
       FILE SECTION.
      *------------*
       FD  LOB-FILE-IN.
       01  LOB-RECORD-IN       PIC X(80).

       WORKING-STORAGE SECTION.
      * HOST VARIABLES

       01 SW-EOF           PIC X  VALUE 'N'.
          88 EOF                  VALUE 'Y'.
       01  INDICATOR PIC S9(4) COMP.
       01  RET-CODE  PIC S999 SIGN LEADING SEPARATE.

      * SQL DECLARES
           EXEC SQL
             INCLUDE SQLCA
           END-EXEC.

           EXEC SQL DECLARE TBTLOB TABLE
           ( PNO                          SMALLINT NOT NULL,
             LOBCOL1                      CLOB(102400),
             TBTLOBROWID                   ROWID NOT NULL,
```

```
       LOBCOL2                         BLOB(204800)
     ) END-EXEC.

 01  DCLTBTLOB.
     10 PNO                   PIC S9(4) USAGE COMP.
* GEWONE CLOB HOST VARIABELE
     10 LOBCOL1              USAGE SQL TYPE IS CLOB(100K).
* CLOB LOCATOR
     10 LOBCOL1-LOCATOR      USAGE SQL TYPE IS CLOB-LOCATOR.


 PROCEDURE DIVISION.
 PROG-START.
     PERFORM INIT
     PERFORM PROCES
     PERFORM INSERT-LOB-VALUE
     PERFORM TERM
     STOP RUN
     .

 INIT.
* INITIALISATIE VAN DE LOCATOR

     EXEC SQL
        SET :LOBCOL1-LOCATOR = ''
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE SQLCODE TO RET-CODE
        DISPLAY 'FOUT BIJ SET LOCATOR INITIALISATIE '
        DISPLAY 'SQLCODE = ' RET-CODE
     END-IF
     .

 PROCES.
*INLEZEN VAN FILE IN DE CLOB VARIABELE  TOT LOB-LENGTH = 800

     OPEN INPUT LOB-FILE-IN
     MOVE 0 TO LOBCOL1-LENGTH
     READ LOB-FILE-IN
        AT END SET EOF TO TRUE
     END-READ
     PERFORM BUILD-LOBCOLUMN UNTIL EOF
* ALS ER NOG RECORDS GELEZEN ZIJN DIE NOG NIET MET EEN APPEND ZIJN
* TOEGEVOEGD AAN DE LOCATOR ---> NOG EEN LAATSTE APPEND
*
     IF LOBCOL1-LENGTH > 0
       PERFORM APPEND-LOCATOR
     END-IF
     .

 BUILD-LOBCOLUMN.
* WEGSCHRIJVEN VAN HET RECORD DAT JUIST GELEZEN IS IN DE
* HOSTVARIABELE LOBCOL1-DATA
* STEL DAT DE TOTALE GROOTTE VAN EEN VARIABELE IN DEZE OMGEVING
* SLECHTS 800 BYTES KAN ZIJN
* NA 10DE RECORD WORDT EEN APPEND AAN DE LOCATOR GEDAAN
** IN WERKELIJKHEID SUPPORTEERT COBOL VOOR OS/390 VARIABELEN
** MET EEN MAXIMALE GROOTTE VAN 16MB
*

     MOVE LOB-RECORD-IN
         TO LOBCOL1-DATA(LOBCOL1-LENGTH + 1 : 80)
     ADD 80 TO LOBCOL1-LENGTH
```

```
           IF LOBCOL1-LENGTH = 800 THEN
              PERFORM APPEND-LOCATOR
              MOVE 0 TO    LOBCOL1-LENGTH
              MOVE ' ' TO  LOBCOL1-DATA
           END-IF

           READ LOB-FILE-IN
             AT END SET EOF TO TRUE
           END-READ
           .

       APPEND-LOCATOR.
      * NA HET 10DE RECORD WORDT DE WAARDE DIE MOMENTEEL
      * IN DE HOSTVARIABELE AANWEZIG IS WEGGESCHREVEN NAAR EEN
      * DATASPACE. DE LOCATOR DIENT OM DE WAARDE TERUG TE VINDEN
      *
           EXEC SQL
               SET :LOBCOL1-LOCATOR =
               CONCAT (:LOBCOL1-LOCATOR, :LOBCOL1)
           END-EXEC

           IF SQLCODE NOT = 0
              MOVE SQLCODE TO RET-CODE
              DISPLAY 'ERROR BIJ SET LOCATOR NA APPEND'
              DISPLAY 'SQLCODE IS : ' RET-CODE
           END-IF
           .
       INSERT-LOB-VALUE.
      * OPVULLEN VAN DE REST VAN DE KOLOMMEN VOOR PNO = 2
           MOVE 2 TO PNO

      *  INSERTEN VAN  DE CLOB-VARIABELE IN DE CLOB KOLOM
           EXEC SQL
              INSERT INTO TBTLOB
              (PNO,
               LOBCOL1)
            VALUES
              (:PNO,
               :LOBCOL1-LOCATOR)
             END-EXEC
           IF SQLCODE NOT = 0
              MOVE SQLCODE TO RET-CODE
              DISPLAY 'FOUT BIJ INSERT'
              DISPLAY 'DE SQLCODE = ' RET-CODE
           END-IF
           .

       TERM.
           CLOSE LOB-FILE-IN
           .

**************
* EXAMPLE   3*
**************

       IDENTIFICATION DIVISION.
      ********************************************************
      * PROGRAMMMA OM EEN SELECT TE DOEN VAN EEN LOB KOLOM
      * DE GEBRUIKTE TABEL BEVAT TWEE LOB KOLOMMEN
      *     - LOBCOL1 VAN HET TYPE CLOB(100K)
      *     - LOBCOL2 VAN HET TYPE BLOB(200K)
      * EN OOK EEN ROWID KOLOM (VERPLICHT EN GENERATED
      *                                      ALWAYS)
```

```
* DE COMPLETE LOBVALUE WORDT IN DE HOST VARIABELE
* GEMATERIALISEERD (APPLICATIE)
* OF ER WORDT GEBRUIKT GEMAAKT VAN EEN LOCATOR
* MATERIALISATIE IN DE DATASPACE
************************************************************
 PROGRAM-ID.         LOBEX3.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
* HOST VARIABLES
 01  INDICATOR PIC S9(4) COMP.
 01  LOBCOL1-LENGTH-OUT    PIC S9(9) SIGN LEADING SEPARATE.
 01  LOBCOL1-DATA-OUT       PIC X(50) VALUE ALL SPACES.
 01  LOB-DATA               PIC X(50) VALUE ALL SPACES.
 01  FUNC-LENGTH           PIC S9(9) COMP.
 01  FUNC-LENGTH-OUT      PIC S9(9) SIGN LEADING SEPARATE.
 77  RET-CODE  PIC S999 SIGN LEADING SEPARATE.

* SQL DECLARES
     EXEC SQL
       INCLUDE SQLCA
     END-EXEC.

     EXEC SQL DECLARE TBTLOB TABLE
     ( PNO                           SMALLINT NOT NULL,
       LOBCOL1                       CLOB(102400),
       TBTLOBROWID                    ROWID NOT NULL,
       LOBCOL2                       BLOB(204800)
     ) END-EXEC.

 01  DCLTBTLOB.
     10 PNO                  PIC S9(4) USAGE COMP.
     10 LOBCOL1              USAGE SQL TYPE IS CLOB (100K).
     10 LOBCOL1-LOCATOR      USAGE SQL TYPE IS CLOB-LOCATOR.


 PROCEDURE DIVISION.
 PROG-START.
     PERFORM SELECT-LOB-IN-HOST-VARIABLE
     PERFORM FUNCTIONS-WITH-HOST-VARIABLE
     PERFORM SELECT-LOB-IN-LOCATOR
     PERFORM FUNCTIONS-WITH-LOB-LOCATOR
     PERFORM TERM
     STOP RUN
     .
 SELECT-LOB-IN-HOST-VARIABLE.
     DISPLAY '****************************'
     DISPLAY 'SELECT-LOB-IN-HOST-VARIABLE'
     DISPLAY '****************************'
     MOVE 1 TO PNO
     EXEC SQL
        SELECT PNO,LOBCOL1
        INTO :PNO, :LOBCOL1
        FROM TBTLOB
        WHERE PNO = :PNO
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE SQLCODE TO RET-CODE
        DISPLAY 'ERROR BIJ SELECT LOB IN HOST VARIABELE'
        DISPLAY 'DE SQLCODE IS : ' RET-CODE
     END-IF
     MOVE LOBCOL1-DATA TO LOBCOL1-DATA-OUT
     DISPLAY 'DE LOB IS '  LOBCOL1-DATA-OUT
```

```
    MOVE LOBCOL1-LENGTH TO LOBCOL1-LENGTH-OUT
    DISPLAY 'DE LOB-LENGTH IS '  LOBCOL1-LENGTH-OUT
    .

FUNCTIONS-WITH-HOST-VARIABLE.
    EXEC SQL
       SELECT SUBSTR(:LOBCOL1, 1, 30)
       INTO :LOB-DATA
       FROM TBTLOB
       WHERE PNO = :PNO
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ SUBSTR HOST-VARIABLE'
       DISPLAY 'DE SQLCODE IS : ' RET-CODE
    END-IF
    DISPLAY 'DE SUBSTR 1-30 LOB-DATA : ' LOB-DATA

    EXEC SQL
       SELECT LENGTH(LOBCOL1)
       INTO :FUNC-LENGTH
       FROM TBTLOB
       WHERE PNO = 1
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ LENGTH HOST-VARIABLE'
       DISPLAY 'DE SQLCODE IS : ' RET-CODE
    END-IF
    MOVE FUNC-LENGTH TO FUNC-LENGTH-OUT
    DISPLAY 'DE LENGTE VAN DE LOB KOLOM IS : ' FUNC-LENGTH-OUT
    .
SELECT-LOB-IN-LOCATOR.
    DISPLAY '***************************'
    DISPLAY 'SELECT-LOB-IN-LOCATOR'
    DISPLAY '***************************'
    MOVE 0 TO LOBCOL1-LENGTH
    MOVE SPACES TO LOBCOL1-DATA
    EXEC SQL
        SET :LOBCOL1-LOCATOR = ''
    END-EXEC
    MOVE 2 TO PNO
    EXEC SQL
       SELECT PNO,LOBCOL1
       INTO :PNO, :LOBCOL1-LOCATOR
       FROM TBTLOB
       WHERE PNO = :PNO
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ SELECT LOB IN LOCATOR'
       DISPLAY 'DE SQLCODE IS : ' RET-CODE
    END-IF
    DISPLAY 'DE LOB-LOCATOR IS '  LOBCOL1-LOCATOR
    MOVE LOBCOL1-DATA TO LOBCOL1-DATA-OUT
    DISPLAY 'DE LOB-DATA IS '  LOBCOL1-DATA-OUT
    MOVE LOBCOL1-LENGTH TO LOBCOL1-LENGTH-OUT
    DISPLAY 'DE LOB-LENGTH IS '  LOBCOL1-LENGTH-OUT

    .
FUNCTIONS-WITH-LOB-LOCATOR.
    DISPLAY '***************************'
    DISPLAY 'FUNCTIONS-WITH-LOB-LOCATOR'
```

```
           DISPLAY '***************************'
           EXEC SQL
              SELECT SUBSTR(:LOBCOL1-LOCATOR, 1, 50)
              INTO :LOB-DATA
              FROM TBTLOB
              WHERE PNO = 2
           END-EXEC
           IF SQLCODE NOT = 0
              MOVE SQLCODE TO RET-CODE
              DISPLAY 'ERROR BIJ SUBSTR LOCATOR'
              DISPLAY 'DE SQLCODE IS : ' RET-CODE
           END-IF
           DISPLAY 'DE LOB-LOCATOR IS '  LOBCOL1-LOCATOR
           DISPLAY 'DE LOB-DATA IS '  LOB-DATA

           MOVE 0 TO FUNC-LENGTH
                     FUNC-LENGTH-OUT
           EXEC SQL
              SELECT LENGTH(:LOBCOL1-LOCATOR)
              INTO :FUNC-LENGTH
              FROM TBTLOB
              WHERE PNO = 2
           END-EXEC
           IF SQLCODE NOT = 0
              MOVE SQLCODE TO RET-CODE
              DISPLAY 'ERROR BIJ LENGTH LOCATOR'
              DISPLAY 'DE SQLCODE IS : ' RET-CODE
           END-IF
           MOVE FUNC-LENGTH TO FUNC-LENGTH-OUT
           DISPLAY 'DE LENGTE VAN DE LOB KOLOM IS : ' FUNC-LENGTH-OUT
           .

       TERM.
           EXEC SQL
              FREE LOCATOR :LOBCOL1-LOCATOR
           END-EXEC
           IF SQLCODE NOT = 0
              MOVE SQLCODE TO RET-CODE
              DISPLAY 'ERROR BIJ FREE LOCATOR'
              DISPLAY 'DE SQLCODE IS : ' RET-CODE
           END-IF
           .


**************
* EXAMPLE  4 *
**************

       IDENTIFICATION DIVISION.
      ********************************************************
      * PROGRAMMMA OM EEN TWEE LOB-VALUES AAN ELKAAR TE CONCATENEREN
      * DE GEBRUIKTE TABEL BEVAT TWEE LOB KOLOMMEN
      *     - LOBCOL1 VAN HET TYPE CLOB(100K)
      *     - LOBCOL2 VAN HET TYPE BLOB(200K)
      * EN OOK EEN ROWID KOLOM (VERPLICHT EN GENERATED
      *                                    ALWAYS)
      * ER WORDT GEBRUIKT GEMAAKT VAN EEN CONCATENATIE VAN LOCATORS
      ********************************************************
       PROGRAM-ID.         LOBEX4.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
      * HOST VARIABLES
```

```
 01  LOB-DATA              PIC X(50) VALUE ALL SPACES.
 01  LOB-LENGTH           PIC S9(9) COMP VALUE 0.
 01  LOB-LENGTH-TOT       PIC S9(9) COMP VALUE 0.
 77  RET-CODE  PIC S999 SIGN LEADING SEPARATE.

* SQL DECLARES
     EXEC SQL
       INCLUDE SQLCA
     END-EXEC.

     EXEC SQL DECLARE TBTLOB TABLE
     ( PNO                            SMALLINT NOT NULL,
       LOBCOL1                        CLOB(102400),
       TBTLOBROWID                     ROWID NOT NULL,
       LOBCOL2                        BLOB(204800)
     ) END-EXEC.

 01  DCLTBTLOB.
     10 PNO                  PIC S9(4) USAGE COMP.
     10 LOBCOL1-LOCATOR1     USAGE SQL TYPE IS CLOB-LOCATOR.
     10 LOBCOL1-LOCATOR2     USAGE SQL TYPE IS CLOB-LOCATOR.
     10 LOBCOL1-LOCATOR3     USAGE SQL TYPE IS CLOB-LOCATOR.


 PROCEDURE DIVISION.
 PROG-START.
     PERFORM SELECT-LOB1
     PERFORM SELECT-LOB2
     PERFORM INSERT-LOB
     PERFORM TERM
     STOP RUN
     .
 SELECT-LOB1.
     DISPLAY '****************************'
     DISPLAY 'SELECT-LOB1'
     DISPLAY '****************************'
     MOVE 0 TO LOB-LENGTH
     MOVE SPACES TO LOB-DATA
     EXEC SQL
         SET :LOBCOL1-LOCATOR1 = ''
     END-EXEC
     MOVE 1 TO PNO
     EXEC SQL
        SELECT LENGTH(LOBCOL1),LOBCOL1
        INTO :LOB-LENGTH, :LOBCOL1-LOCATOR1
        FROM TBTLOB
        WHERE PNO = :PNO
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE SQLCODE TO RET-CODE
        DISPLAY 'ERROR BIJ SELECT LOB1'
        DISPLAY 'DE SQLCODE IS : ' RET-CODE
     END-IF

     .
 SELECT-LOB2.
     DISPLAY '****************************'
     DISPLAY 'SELECT-LOB2'
     DISPLAY '****************************'
     MOVE 0 TO LOB-LENGTH
     MOVE SPACES TO LOB-DATA
     EXEC SQL
```

```
            SET :LOBCOL1-LOCATOR2 = ''
        END-EXEC
        MOVE 2 TO PNO
        EXEC SQL
           SELECT LENGTH(LOBCOL1),LOBCOL1
           INTO :LOB-LENGTH, :LOBCOL1-LOCATOR2
           FROM TBTLOB
           WHERE PNO = :PNO
        END-EXEC
        IF SQLCODE NOT = 0
           MOVE SQLCODE TO RET-CODE
           DISPLAY 'ERROR BIJ SELECT LOB2'
           DISPLAY 'DE SQLCODE IS : ' RET-CODE
        END-IF
        .

    INSERT-LOB.
        DISPLAY '****************************'
        DISPLAY 'INSERT-LOB'
        DISPLAY '****************************'
        EXEC SQL
            SET :LOBCOL1-LOCATOR3 =
             CONCAT(:LOBCOL1-LOCATOR1,:LOBCOL1-LOCATOR2)
        END-EXEC
        IF SQLCODE NOT = 0
           MOVE SQLCODE TO RET-CODE
           DISPLAY 'ERROR BIJ SET LOCATOR 3'
           DISPLAY 'DE SQLCODE IS : ' RET-CODE
        END-IF

        EXEC SQL
            INSERT INTO TBTLOB
            (PNO,LOBCOL1)
            VALUES
            (5,
             :LOBCOL1-LOCATOR3)
        END-EXEC
        IF SQLCODE NOT = 0
           MOVE SQLCODE TO RET-CODE
           DISPLAY 'ERROR BIJ INSERT'
           DISPLAY 'DE SQLCODE IS : ' RET-CODE
        END-IF
        .
    TERM.
        EXEC SQL
           FREE LOCATOR :LOBCOL1-LOCATOR1,:LOBCOL1-LOCATOR2,
                        :LOBCOL1-LOCATOR3
        END-EXEC
        IF SQLCODE NOT = 0
           MOVE SQLCODE TO RET-CODE
           DISPLAY 'ERROR BIJ FREE LOCATORS'
           DISPLAY 'DE SQLCODE IS : ' RET-CODE
        END-IF
        .


**************
* EXAMPLE  5 *
**************

       IDENTIFICATION DIVISION.
      ********************************************************
     * PROGRAMMMA OM EEN DEEL VAN EEN CLOB TE BEWAREN IN EEN VARIABELE
```

```
      * DE GEBRUIKTE TABEL BEVAT TWEE LOB KOLOMMEN
      *     - LOBCOL1 VAN HET TYPE CLOB(100K)
      *     - LOBCOL2 VAN HET TYPE BLOB(200K)
      * EN OOK EEN ROWID KOLOM (VERPLICHT EN GENERATED
      *                                    ALWAYS)
      * ER WORDT GEBRUIKT GEMAAKT VAN LOCATORS
      ************************************************************
       PROGRAM-ID.         LOBEX5.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
      * HOST VARIABLES

       01  LOB-DATA              PIC X(50) VALUE ALL SPACES.
       01  LOB-BEGIN             PIC S9(9) COMP VALUE 0.
       01  LOB-END               PIC S9(9) COMP VALUE 0.
       01  LOB-LENGTH            PIC S9(9) COMP VALUE 0.
       77  RET-CODE              PIC S999 SIGN LEADING SEPARATE.

      * SQL DECLARES
           EXEC SQL
             INCLUDE SQLCA
           END-EXEC.

           EXEC SQL DECLARE TBTLOB TABLE
           ( PNO                            SMALLINT NOT NULL,
             LOBCOL1                        CLOB(102400),
             TBTLOBROWID                     ROWID NOT NULL,
             LOBCOL2                        BLOB(204800)
           ) END-EXEC.

*****************************************************************
     * COBOL DECLARATION FOR TABLE TBTLOB                          *
*****************************************************************
       01  DCLTBTLOB.
           10 PNO                  PIC S9(4) USAGE COMP.
           10 LOBCOL1-LOCATOR1     USAGE SQL TYPE IS CLOB-LOCATOR.
           10 LOBCOL1-LOCATOR2     USAGE SQL TYPE IS CLOB-LOCATOR.


       PROCEDURE DIVISION.
       PROG-START.
           PERFORM SET-LOCATOR
           PERFORM SELECT-BEGIN-OF-TEXT
           PERFORM SELECT-END-OF-TEXT
           PERFORM INSERT-LOB
           PERFORM TERM
           STOP RUN
           .
       SET-LOCATOR.
           DISPLAY '****************************'
           DISPLAY 'SETLOCATOR'
           DISPLAY '****************************'
           EXEC SQL
               SET :LOBCOL1-LOCATOR1 = ''
           END-EXEC
           MOVE 1 TO PNO
           EXEC SQL
              SELECT LOBCOL1
              INTO :LOBCOL1-LOCATOR1
              FROM TBTLOB
              WHERE PNO = :PNO
```

```
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ SELECT LOCATOR'
       DISPLAY 'DE SQLCODE IS : ' RET-CODE
    END-IF

    .
SELECT-BEGIN-OF-TEXT.
    DISPLAY '****************************'
    DISPLAY 'SELECT-BEGIN-OF-TEXT'
    DISPLAY '****************************'
    EXEC SQL
        SET :LOB-BEGIN =
            POSSTR(:LOBCOL1-LOCATOR1,'DATA DIVISION')
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ SELECT BEGIN'
       DISPLAY 'DE SQLCODE IS : ' RET-CODE
    END-IF
    .

SELECT-END-OF-TEXT.
    DISPLAY '****************************'
    DISPLAY 'SELECT-END-OF-TEXT'
    DISPLAY '****************************'
    EXEC SQL
        SET :LOB-END =
            POSSTR(:LOBCOL1-LOCATOR1,'PROCEDURE DIVISION')
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ SELECT END'
       DISPLAY 'DE SQLCODE IS : ' RET-CODE
    END-IF
    .

INSERT-LOB.
    DISPLAY '****************************'
    DISPLAY 'INSERT-LOB'
    DISPLAY '****************************'
    EXEC SQL
        SET :LOBCOL1-LOCATOR2 =
         SUBSTR(:LOBCOL1-LOCATOR1,:LOB-BEGIN,
                :LOB-END - :LOB-BEGIN)
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ SET LOCATOR 2'
       DISPLAY 'DE SQLCODE IS : ' RET-CODE
    END-IF

    EXEC SQL
        INSERT INTO TBTLOB
        (PNO,LOBCOL1)
        VALUES
        (6,
         :LOBCOL1-LOCATOR2)
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ INSERT'
```

```
           DISPLAY 'DE SQLCODE IS : ' RET-CODE
       END-IF
       .
   TERM.
       EXEC SQL
          FREE LOCATOR :LOBCOL1-LOCATOR1,:LOBCOL1-LOCATOR2
       END-EXEC
       IF SQLCODE NOT = 0
          MOVE SQLCODE TO RET-CODE
          DISPLAY 'ERROR BIJ FREE LOCATORS'
          DISPLAY 'DE SQLCODE IS : ' RET-CODE
       END-IF
       .


**************
* EXAMPLE 6 *
**************

    IDENTIFICATION DIVISION.
   ***********************************************************
   * PROGRAMMMA OM EEN DEEL VAN EEN CLOB TE DELETEN
   * DE GEBRUIKTE TABEL BEVAT TWEE LOB KOLOMMEN
   *     - LOBCOL1 VAN HET TYPE CLOB(100K)
   *     - LOBCOL2 VAN HET TYPE BLOB(200K)
   * EN OOK EEN ROWID KOLOM (VERPLICHT EN GENERATED
   *                                    ALWAYS)
   * ER WORDT GEBRUIKT GEMAAKT VAN LOCATORS
   * RIJ MET PNO = 7 WORDT GEWIJZIGD. DEZE RIJ IS EEN EXACTE COPY
   * VAN DE RIJ MET NR 1
   * HET STUK TEKST DAT WORDT VERWIJDERD IS HET STUK DAT TIJDENS
   * DE UITVOERING VAN LOBEX5 IN DE RIJ MET PNO = 6
   * GEINSERT IS.
   ***********************************************************
    PROGRAM-ID.         LOBEX6.
    ENVIRONMENT DIVISION.
    DATA DIVISION.
    WORKING-STORAGE SECTION.
   * HOST VARIABLES

    01  LOB-DATA               PIC X(50) VALUE ALL SPACES.
    01  LOB-BEGIN              PIC S9(9) COMP VALUE 0.
    01  LOB-END                PIC S9(9) COMP VALUE 0.
    01  LOB-LENGTH             PIC S9(9) COMP VALUE 0.
    77  RET-CODE               PIC S999 SIGN LEADING SEPARATE.

   * SQL DECLARES
       EXEC SQL
         INCLUDE SQLCA
       END-EXEC.

       EXEC SQL DECLARE TBTLOB TABLE
       ( PNO                           SMALLINT NOT NULL,
         LOBCOL1                       CLOB(102400),
         TBTLOBROWID                    ROWID NOT NULL,
         LOBCOL2                       BLOB(204800)
       ) END-EXEC.

    01  DCLTBTLOB.
       10 PNO                  PIC S9(4) USAGE COMP.
       10 LOBCOL1-LOCATOR1     USAGE SQL TYPE IS CLOB-LOCATOR.
       10 LOBCOL1-LOCATOR2     USAGE SQL TYPE IS CLOB-LOCATOR.
```

```
PROCEDURE DIVISION.
PROG-START.
    PERFORM SET-LOCATOR
    PERFORM SELECT-BEGIN-OF-TEXT
    PERFORM SELECT-END-OF-TEXT
    PERFORM DELETE-PART-OF-LOB
    PERFORM TERM
    STOP RUN
    .
SET-LOCATOR.
    DISPLAY '****************************'
    DISPLAY 'SETLOCATOR'
    DISPLAY '****************************'
    EXEC SQL
        SET :LOBCOL1-LOCATOR1 = ''
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ SET LOCATOR'
       DISPLAY 'DE SQLCODE IS : ' RET-CODE
    END-IF
    MOVE 7 TO PNO
    EXEC SQL
       SELECT LOBCOL1
       INTO :LOBCOL1-LOCATOR1
       FROM TBTLOB
       WHERE PNO = :PNO
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ SELECT LOCATOR'
       DISPLAY 'DE SQLCODE IS : ' RET-CODE
    END-IF
    .
SELECT-BEGIN-OF-TEXT.
    DISPLAY '****************************'
    DISPLAY 'SELECT-BEGIN-OF-TEXT'
    DISPLAY '****************************'
    EXEC SQL
        SET :LOB-BEGIN =
            POSSTR(:LOBCOL1-LOCATOR1,'DATA DIVISION')
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ SELECT BEGIN'
       DISPLAY 'DE SQLCODE IS : ' RET-CODE
    END-IF
    .

SELECT-END-OF-TEXT.
    DISPLAY '****************************'
    DISPLAY 'SELECT-END-OF-TEXT'
    DISPLAY '****************************'
    EXEC SQL
        SET :LOB-END =
            POSSTR(:LOBCOL1-LOCATOR1,'PROCEDURE DIVISION')
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ SELECT END'
       DISPLAY 'DE SQLCODE IS : ' RET-CODE
```

```
           END-IF
           .

       DELETE-PART-OF-LOB.
           DISPLAY '****************************'
           DISPLAY 'DELETE PART OF LOB '
           DISPLAY '****************************'
           EXEC SQL
               SET :LOBCOL1-LOCATOR2 =
                 SUBSTR(:LOBCOL1-LOCATOR1, 1 , :LOB-BEGIN - 1)
                 CONCAT SUBSTR(:LOBCOL1-LOCATOR1, :LOB-END)
           END-EXEC
           IF SQLCODE NOT = 0
              MOVE SQLCODE TO RET-CODE
              DISPLAY 'ERROR BIJ SET LOCATOR 2'
              DISPLAY 'DE SQLCODE IS : ' RET-CODE
           END-IF

           EXEC SQL
               UPDATE TBTLOB
               SET LOBCOL1 = :LOBCOL1-LOCATOR2
               WHERE PNO = :PNO
           END-EXEC
           IF SQLCODE NOT = 0
              MOVE SQLCODE TO RET-CODE
              DISPLAY 'ERROR BIJ UPDATE'
              DISPLAY 'DE SQLCODE IS : ' RET-CODE
           END-IF
           .
       TERM.
           EXEC SQL
              FREE LOCATOR :LOBCOL1-LOCATOR1,:LOBCOL1-LOCATOR2
           END-EXEC
           IF SQLCODE NOT = 0
              MOVE SQLCODE TO RET-CODE
              DISPLAY 'ERROR BIJ FREE LOCATORS'
              DISPLAY 'DE SQLCODE IS : ' RET-CODE
           END-IF
           .


**************
* EXAMPLE  7 *
**************

       IDENTIFICATION DIVISION.
      ********************************************************
      * PROGRAMMMA OM EEN DEEL VAN EEN CLOB TE UPDATEN
      * DE GEBRUIKTE TABEL BEVAT TWEE LOB KOLOMMEN
      *     - LOBCOL1 VAN HET TYPE CLOB(100K)
      *     - LOBCOL2 VAN HET TYPE BLOB(200K)
      * EN OOK EEN ROWID KOLOM (VERPLICHT EN GENERATED
      *                                       ALWAYS)
      * ER WORDT GEBRUIKT GEMAAKT VAN LOCATORS
      * RIJ MET PNO = 8 WORDT GEWIJZIGD. DEZE RIJ IS EEN EXACTE COPY
      * VAN DE RIJ MET NR 1
      * HET STUK TEKST DAT WORDT GEUPDATED IS HET STUK DAT TIJDENS
      * DE UITVOERING VAN LOBEX5 IN DE RIJ MET PNO = 6
      * GEINSERT IS.
      ********************************************************
       PROGRAM-ID.         LOBEX7.
       ENVIRONMENT DIVISION.
```

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*HOST VARIABLES
 01  INDICATOR PIC S9(4) COMP.
 01  LOB-DATA               PIC X(50) VALUE ALL SPACES.
 01  LOB-BEGIN              PIC S9(9) COMP VALUE 0.
 01  LOB-END                PIC S9(9) COMP VALUE 0.
 01  LOB-LENGTH             PIC S9(9) COMP VALUE 0.
 01  NEW-TEXT               PIC X(80).
 77  RET-CODE               PIC S999 SIGN LEADING SEPARATE.

* SQL DECLARES
     EXEC SQL
       INCLUDE SQLCA
     END-EXEC.

     EXEC SQL DECLARE TBTLOB TABLE
     ( PNO                           SMALLINT NOT NULL,
       LOBCOL1                       CLOB(102400),
       TBTLOBROWID                    ROWID NOT NULL,
       LOBCOL2                       BLOB(204800)
     ) END-EXEC.

 01  DCLTBTLOB.
     10 PNO                  PIC S9(4) USAGE COMP.
     10 LOBCOL1-LOCATOR1     USAGE SQL TYPE IS CLOB-LOCATOR.
     10 LOBCOL1-LOCATOR2     USAGE SQL TYPE IS CLOB-LOCATOR.


 PROCEDURE DIVISION.
 PROG-START.
     PERFORM SET-LOCATOR
     PERFORM SELECT-BEGIN-OF-TEXT
     PERFORM SELECT-END-OF-TEXT
     PERFORM UPDATE-LOB
     PERFORM TERM
     STOP RUN
     .
 SET-LOCATOR.
     DISPLAY '****************************'
     DISPLAY 'SET LOCATOR'
     DISPLAY '****************************'
     EXEC SQL
         SET :LOBCOL1-LOCATOR1 = ''
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE SQLCODE TO RET-CODE
        DISPLAY 'ERROR BIJ SET LOCATOR'
        DISPLAY 'DE SQLCODE IS : ' RET-CODE
     END-IF

     MOVE 8 TO PNO
     EXEC SQL
        SELECT LOBCOL1
        INTO :LOBCOL1-LOCATOR1
        FROM TBTLOB
        WHERE PNO = :PNO
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE SQLCODE TO RET-CODE
        DISPLAY 'ERROR BIJ SELECT LOCATOR'
        DISPLAY 'DE SQLCODE IS : ' RET-CODE
     END-IF
```

```
    .
SELECT-BEGIN-OF-TEXT.
    DISPLAY '****************************'
    DISPLAY 'SELECT-BEGIN-OF-TEXT'
    DISPLAY '****************************'
    EXEC SQL
        SET :LOB-BEGIN =
            POSSTR(:LOBCOL1-LOCATOR1,'DATA DIVISION')
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ SELECT BEGIN'
       DISPLAY 'DE SQLCODE IS : ' RET-CODE
    END-IF
    .

SELECT-END-OF-TEXT.
    DISPLAY '****************************'
    DISPLAY 'SELECT-END-OF-TEXT'
    DISPLAY '****************************'
    EXEC SQL
        SET :LOB-END =
            POSSTR(:LOBCOL1-LOCATOR1,'PROCEDURE DIVISION')
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ SELECT END'
       DISPLAY 'DE SQLCODE IS : ' RET-CODE
    END-IF
    .

UPDATE-LOB.
    DISPLAY '****************************'
    DISPLAY 'UPDATE-LOB'
    DISPLAY '****************************'
    MOVE 'DIT IS DE UPDATE TEXT' TO NEW-TEXT
    EXEC SQL
        SET :LOBCOL1-LOCATOR2 =
          SUBSTR(:LOBCOL1-LOCATOR1, 1 , :LOB-BEGIN - 1)
          CONCAT :NEW-TEXT
          CONCAT SUBSTR(:LOBCOL1-LOCATOR1, :LOB-END)
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ SET LOCATOR 2'
       DISPLAY 'DE SQLCODE IS : ' RET-CODE
    END-IF

    EXEC SQL
        UPDATE TBTLOB
        SET LOBCOL1 = :LOBCOL1-LOCATOR2
        WHERE PNO = :PNO
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ UPDATE'
       DISPLAY 'DE SQLCODE IS : ' RET-CODE
    END-IF
    .
TERM.
    EXEC SQL
       FREE LOCATOR :LOBCOL1-LOCATOR1,:LOBCOL1-LOCATOR2
    END-EXEC
```

```
           IF SQLCODE NOT = 0
              MOVE SQLCODE TO RET-CODE
              DISPLAY 'ERROR BIJ FREE LOCATORS'
              DISPLAY 'DE SQLCODE IS : ' RET-CODE
           END-IF
           .


**************
* EXAMPLE  8 *
**************

       IDENTIFICATION DIVISION.
      ********************************************************
      * PROGRAMMMA OM EEN CLOB TE UNLOADEN IN EEN DATASET
      * DE GEBRUIKTE TABEL BEVAT TWEE LOB KOLOMMEN
      *     - LOBCOL1 VAN HET TYPE CLOB(100K)
      *     - LOBCOL2 VAN HET TYPE BLOB(200K)
      * EN OOK EEN ROWID KOLOM (VERPLICHT EN GENERATED
      *                                      ALWAYS)
      * DE CLOB VALUE WORDT IN ZIJN GEHEEL BINNENGEBRACHT IN DE
      * APPLICATIE IN EEN HOST VARIABELE
      * DIT KAN ENKEL WANNEER DE CLOB VALUE NIET TE GROOT IS
      * DEZE VARIABELE WORDT DAN VOLGENS DE JUISTE RECORDLENGTE
      * RECORD PER RECORD NAAR EEN OUTPUT FILE WEGGESCHREVEN
      ********************************************************
       PROGRAM-ID.         LOBEX8.
       ENVIRONMENT DIVISION.
       INPUT-OUTPUT SECTION.
      *--------------------*
       FILE-CONTROL.
      *------------*
           SELECT LOB-FILE-OUT  ASSIGN TO FILEOUT.

       DATA DIVISION.
       FILE SECTION.
      *------------*
       FD  LOB-FILE-OUT.
       01  LOB-RECORD-OUT      PIC X(80).

       WORKING-STORAGE SECTION.
      * HOST VARIABLES
       01 SW-EOF           PIC X  VALUE 'N'.
          88 EOF                  VALUE 'Y'.
       01  INDICATOR PIC S9(4) COMP.
       01  RECORD-OUT-LENGTH      PIC S9(4) COMP.
       01  LOB-COUNTER            PIC S9(9) COMP.
       01  RECORD-COUNTER         PIC S9(4) COMP.
       77  RET-CODE               PIC S999 SIGN LEADING SEPARATE.

      * SQL DECLARES
           EXEC SQL
             INCLUDE SQLCA
           END-EXEC.

           EXEC SQL DECLARE TBTLOB TABLE
           ( PNO                          SMALLINT NOT NULL,
             LOBCOL1                      CLOB(102400),
             TBTLOBROWID                   ROWID NOT NULL,
             LOBCOL2                      BLOB(204800)
           ) END-EXEC.

       01  DCLTBTLOB.
```

```
           10 PNO                  PIC S9(4) USAGE COMP.
           10 LOBCOL1              USAGE SQL TYPE IS CLOB (100K).

       PROCEDURE DIVISION.
       PROG-START.
           PERFORM SELECT-LOB
           PERFORM WRITE-INIT
          PERFORM WRITE-LOB-TO-FILE UNTIL LOB-COUNTER > LOBCOL1-LENGTH
           PERFORM WRITE-TERM
           STOP RUN
           .
       SELECT-LOB.
           DISPLAY '****************************'
           DISPLAY 'SELECT LOB '
           DISPLAY '****************************'

           MOVE 1 TO PNO
           EXEC SQL
              SELECT LOBCOL1
              INTO :LOBCOL1
              FROM TBTLOB
              WHERE PNO = :PNO
           END-EXEC
           IF SQLCODE NOT = 0
              MOVE SQLCODE TO RET-CODE
              DISPLAY 'ERROR BIJ SELECT LOB IN HOST VARIABELE'
              DISPLAY 'DE SQLCODE IS : ' RET-CODE
           END-IF
           .

       WRITE-INIT.
           DISPLAY '****************************'
           DISPLAY 'WRITE-INIT'
           DISPLAY '****************************'
           MOVE 80 TO RECORD-OUT-LENGTH
           MOVE 1 TO LOB-COUNTER
           MOVE 0 TO RECORD-COUNTER
           OPEN OUTPUT LOB-FILE-OUT
           .
       WRITE-LOB-TO-FILE.
           DISPLAY '****************************'
           DISPLAY 'WRITE-LOB-TO-FILE'
           DISPLAY '****************************'
           MOVE LOBCOL1-DATA(LOB-COUNTER : RECORD-OUT-LENGTH)
                TO LOB-RECORD-OUT
           WRITE LOB-RECORD-OUT
           COMPUTE RECORD-COUNTER = RECORD-COUNTER + 1
           COMPUTE LOB-COUNTER = LOB-COUNTER + RECORD-OUT-LENGTH
           .

       WRITE-TERM.
           DISPLAY '****************************************'
           DISPLAY '* EINDE                                 *'
           DISPLAY '****************************************'
           CLOSE LOB-FILE-OUT
           .


**************
* EXAMPLE  9 *
**************

       IDENTIFICATION DIVISION.
```

```
*********************************************************
* PROGRAMMMA OM EEN CLOB TE UNLOADEN IN EEN DATASET
* DE GEBRUIKTE TABEL BEVAT TWEE LOB KOLOMMEN
*     - LOBCOL1 VAN HET TYPE CLOB(100K)
*     - LOBCOL2 VAN HET TYPE BLOB(200K)
* EN OOK EEN ROWID KOLOM (VERPLICHT EN GENERATED
*                                    ALWAYS)
* DE CLOB VALUE WORDT VIA EEN LOCATOR GELOCALISEERD
* ER WORDT VERONDERSTELD DAT DE MAXIMALE GROOTTE VAN EEN
* HOSTVARIABELE 800 BYTES IS
* DEZE VARIABELE WORDT DAN VOLGENS DE JUISTE RECORDLENGTE
* RECORD PER RECORD NAAR EEN OUTPUT FILE WEGGESCHREVEN
*********************************************************
 PROGRAM-ID.         LOBEX9.
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
*--------------------*
 FILE-CONTROL.
*------------*
     SELECT LOB-FILE-OUT  ASSIGN TO FILEOUT.

 DATA DIVISION.
 FILE SECTION.
*------------*
 FD  LOB-FILE-OUT.
 01  LOB-RECORD-OUT      PIC X(80).

 WORKING-STORAGE SECTION.
* HOST VARIABLES
 01 SW-EOF            PIC X  VALUE 'N'.
    88 EOF                   VALUE 'Y'.
 01  LOB-LENGTH              PIC S9(9) COMP.
 01  SUBSTR-COUNTER          PIC S9(9) COMP.
 01  BEGIN-SUBSTR            PIC S9(9) COMP.
 01  NR-OF-SUBSTR            PIC S9(9) COMP.
 01  REMAIN-SUBSTR           PIC S9(9) COMP.
 01  LOB-COUNTER             PIC S9(9) COMP.
 01  RECORD-COUNTER          PIC S9(4) COMP.
 01  LOB-800                 PIC X(800).
 01  LOB-80                  PIC X(80).
 77  RET-CODE                PIC S999 SIGN LEADING SEPARATE.

* SQL DECLARES
     EXEC SQL
       INCLUDE SQLCA
     END-EXEC.

     EXEC SQL DECLARE TBTLOB TABLE
     ( PNO                           SMALLINT NOT NULL,
       LOBCOL1                       CLOB(102400),
       TBTLOBROWID                    ROWID NOT NULL,
       LOBCOL2                       BLOB(204800)
     ) END-EXEC.

 01  DCLTBTLOB.
     10 PNO                  PIC S9(4) USAGE COMP.
     10 LOBCOL1-LOCATOR      USAGE SQL TYPE IS CLOB-LOCATOR.


 PROCEDURE DIVISION.
 PROG-START.
     PERFORM SELECT-LOB-IN-LOCATOR
     PERFORM DETERMINE-NR-OF-SUBSTR
```

```
    PERFORM WRITE-INIT
    PERFORM SUBSTR-LOB-IN-PARTS-OF-800 VARYING
            SUBSTR-COUNTER FROM 1 BY 1
            UNTIL SUBSTR-COUNTER > NR-OF-SUBSTR
    PERFORM WRITE-REMAINING-RECORDS
    PERFORM TERM
    STOP RUN
    .
SELECT-LOB-IN-LOCATOR.
    DISPLAY '****************************'
    DISPLAY 'SELECT-LOB-IN-LOCATOR'
    DISPLAY '****************************'

    MOVE 1 TO PNO
    EXEC SQL
       SELECT LENGTH(LOBCOL1), LOBCOL1
       INTO :LOB-LENGTH, :LOBCOL1-LOCATOR
       FROM TBTLOB
       WHERE PNO = :PNO
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ SELECT LOB IN LOCATOR'
       DISPLAY 'DE SQLCODE IS : ' RET-CODE
    END-IF
    .

DETERMINE-NR-OF-SUBSTR.
    DISPLAY '****************************'
    DISPLAY 'DETERMINE-NR-OF-SUBSTR'
    DISPLAY '****************************'

    COMPUTE NR-OF-SUBSTR = LOB-LENGTH / 800
    COMPUTE REMAIN-SUBSTR = LOB-LENGTH -
                            (NR-OF-SUBSTR * 800)
    MOVE REMAIN-SUBSTR TO REMAIN-SUBSTR-OUT
    DISPLAY 'NR OF REMAINING BYTES IS : '
    DISPLAY REMAIN-SUBSTR-OUT '='
    DISPLAY LOB-LENGTH-OUT '- ( ' NR-OF-SUBSTR-OUT ' * 800)'
    .
SUBSTR-LOB-IN-PARTS-OF-800.
    DISPLAY '****************************'
    DISPLAY 'SUBSTR-LOB-IN-PARTS-OF-800'
    DISPLAY '****************************'

    EXEC SQL
     SET :LOB-800 =
          SUBSTR(:LOBCOL1-LOCATOR,:BEGIN-SUBSTR, 800)
    END-EXEC
    IF SQLCODE NOT = 0
       MOVE SQLCODE TO RET-CODE
       DISPLAY 'ERROR BIJ SET SUBSTR IN LOB'
       DISPLAY 'DE SQLCODE IS : ' RET-CODE
    END-IF
    MOVE 1 TO LOB-COUNTER
    PERFORM WRITE-LOB-TO-FILE UNTIL LOB-COUNTER > 800
    ADD 800 TO BEGIN-SUBSTR
    .
WRITE-INIT.
    DISPLAY '****************************'
    DISPLAY 'WRITE-INIT'
    DISPLAY '****************************'
    MOVE 1 TO BEGIN-SUBSTR
```

```
    OPEN OUTPUT LOB-FILE-OUT
    .
WRITE-LOB-TO-FILE.
    DISPLAY 'WRITE-LOB-TO-FILE'
    MOVE LOB-800(LOB-COUNTER : 80)
         TO LOB-RECORD-OUT
    WRITE LOB-RECORD-OUT
    COMPUTE RECORD-COUNTER = RECORD-COUNTER + 1
    COMPUTE LOB-COUNTER = LOB-COUNTER + 80
    .
WRITE-REMAINING-RECORDS.
    IF REMAIN-SUBSTR > 0
       DISPLAY '****************************'
       DISPLAY 'WRITE-REMAINING-RECORDS'
       DISPLAY '****************************'

       EXEC SQL
          SET :LOB-800 =
          SUBSTR(:LOBCOL1-LOCATOR,:BEGIN-SUBSTR,:REMAIN-SUBSTR)
       END-EXEC
       IF SQLCODE NOT = 0
         MOVE SQLCODE TO RET-CODE
         DISPLAY 'ERROR BIJ SET SUBSTR IN LOB BIJ REMAIN'
         DISPLAY 'DE SQLCODE IS : ' RET-CODE
       END-IF
       MOVE 1 TO LOB-COUNTER
       PERFORM WRITE-LOB-TO-FILE
           UNTIL LOB-COUNTER > REMAIN-SUBSTR
    END-IF
    .
TERM.
    DISPLAY '**************************************'
    DISPLAY '* EINDE                               *'
    DISPLAY '**************************************'
    CLOSE LOB-FILE-OUT
    .
```