102 DB2

April 2001

In this issue

update

DB2 Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office

Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs £255.00 in the UK; $380.00 in the USA and Canada; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1997 issue, are available separately to subscribers for £22.50 ($33.50) each including postage.

DB2 Update on-line

Code from *DB2 Update* can be downloaded from our Web site at http://www.xephon.com/db2update.html; you will need to supply a word from the printed issue.

Editor

Trevor Eddolls

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, and other contents of this journal before making any use of it.

Contributions

When Xephon is given copyright, articles published in *DB2 Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*, or you can download a copy from www.xephon.com/contnote.html.

Using OLAP with DB2

OLAP (On-Line Analytical Processing) is a critical data warehouse function that allows users to get basic strategic information from DB2 without using separate OLAP software or hardware such as DB2 OLAP server for OS/390. For more information on data warehouses see *Using a relational database for data warehouses, DB2 Update*, Issue 83, September 1999 and *A relational schema for a data warehouse, DB2 Update*, Issue 86, December 1999. IBM included OLAP in Version 6 of DB2 UDB and provided corrections and extensions in Version 7.

The syntax including V7 corrections and extensions is:

```
OLAP-function

|--+|  ranking-function     |----+-----|
   +|  numbering-function   |---
   +-| aggregation-function |----'
```

OLAP provides the ability to return ranking, row numbering, and existing column function data as a scalar value in a query result table. Restrictions include:

1 Must be included in expressions in a select-list or ORDER BY clause.

2 The OLAP function is applied to the result table of the innermost subselect containing it.

3 It *cannot* be used as a column function.

```
ranking-function

|--+RANK ()--------+-OVER---------------->
   '-DENSE_RANK ()-'             --DENSERANK is synonym
>----(--+-----------------------------+-->
        '| window-partition-clause |--'
>----| window-order-clause |--)----------|
```

The ranking function computes the ordinal rank of a row OVER a specified *window* (defined below). The rank of a row is 1 plus the number of rows that precede it. Equal rows are assigned the same rank.

Specifying DENSERANK prevents a numeric gap. For example, if there are three equal first rows plus a fourth row then RANK would be 1 1 1 4 whereas DENSERANK would be 1 1 1 2.

```
numbering-function (ROWNUMBER is synonym)

|---ROW_NUMBER()--OVER---(--+----------------------------+-->
                            '| window-partition-clause |--'
>----+-------------------------+--)-----------------------|
     '-| window-order-clause |--'
```

The ROWNUMBER function computes a sequential row number OVER a specified window starting at 1. ORDER BY maintains the sequence – if omitted, then the row numbers are arbitrary as returned by the subselect.

For RANK, DENSERANK, and ROWNUMBER, the data type is BIGINT and cannot be null.

```
aggregation-function

|---column-function()--OVER--(-+----------------------------+-------->
                              '-| window-partition-clause |--'
>----+-------------------------------------------------------------+>
     '-| window-order-clause |-+-----------------------------------+-'
                               '-| window-aggregation-group-clause |-'
```

The aggregation-function allows *column-functions* like AVG, COUNT, MAX, MIN, SUM, etc, OVER a specified window. Window-aggregation-group-clause is only used within an aggregation function.

```
window-partition-clause

                  .-,-------------------------.
                  V                           |
|---PARTITION BY-----partitioning-expression---+-----------------|
```

PARTITION BY specifies the window for the OLAP function; *partitioning-expression* defines the result set partition. Restrictions are:

1 Each *column-name* must unambiguously reference an OLAP function result set column within a subselect.

2 *partitioning-expression* cannot include a scalar-fullselect, undeterministic function, or external action.

3 *partitioning-expression* length < 256 bytes.

```
window-order-clause

              .-,-----------------------------------------.
              V                         .-| asc  option |---.   |
|---ORDER BY-----sort-key-expression--+-------------------+---+---|
                                      '-| desc option |---'
```

ORDER BY defines a row sequence within a partition that determines the value of the OLAP function *or* the ROW meaning (defined below) values in a window-aggregation-group-clause; *sort-key-expression* defines a row sequence within a window partition.

```
asc|desc options

|---ASC|DESC--+-----------------+--------------------------------|
              'NULLS LAST|FIRST-'
```

ASC sorts in ascending sequence; DESC in descending sequence. NULLS FIRST places null values before non-null; NULLS LAST places null values after non-null.

```
window-aggregation-group-clause (only for accumulation-function)

|---+-ROWS--+---+-| group-start |---+----------------------------|
    '-RANGE-'   +-| group-between |-+
                +-| group-end |-----'
```

Window-aggregation-group-clause specifies the aggregation group. ROWS defines by counting rows; RANGE defines by an offset from a sort key.

```
group-start clause

|---+UNBOUNDED PRECEDING-----------+----------------------------|
    +unsigned-constant--PRECEDING--
    '-CURRENT ROW------------------'
```

Group-start specifies a start row with the end row being the current row. It is equivalent to BETWEEN group-start AND CURRENT ROW, where CURRENT ROW is the application group boundary. UNBOUNDED PRECEDING specifies the entire partition preceding the current row; *unsigned-constant* specifies ROWS or a RANGE preceding the current row.

```
     group-between clause

|---BETWEEN--| group-bound1 |--AND--| group-bound2 |--------------|
```

Group-between specifies both the start and end rows based on either ROWS or RANGE.

```
          group-bound clauses

|---+UNBOUNDED PRECEDING|FOLLOWING--+----------------------------|
    +unsigned-constant--PRECEDING---
    +unsigned-constant--FOLLOWING---
    '-CURRENT ROW-------------------'
```

Group-bound1 uses UNBOUNDED PRECEDING; group-bound2 uses FOLLOWING.

```
     group-end clause

|---+-UNBOUNDED FOLLOWING--+--------+----------------------------|
    +-unsigned-constant--FOLLOWING--'
```

Group-end specifies an end row with the start row being the current row. It is equivalent to BETWEEN CURRENT ROW AND group-end.

EXAMPLES

The following logical table is used to illustrate the OLAP queries:

```
INSTRUCTOR(INSTRUCTOR_ID, COLLEGE, surname, annual_salary, edlevel,…)
```

Query 1: rank instructors by annual salary if > $39,000. Sort by surname:

```
SELECT   instructor_id, surname, annual_salary
RANK() OVER (ORDER BY annual_salary DESC) AS rank_salary
FROM     instructor
WHERE    annual_salary > 39000
ORDER BY surname
```

To sort by rank then:

```
ORDER BY rank_salary
```

or by a more complicated:

```
ORDER BY RANK() OVER (ORDER BY salary DESC)
```

Query 2: rank colleges by average annual salary:

```
SELECT   college, AVG(annual_salary)
RANK() OVER (ORDER BY AVG(salary) DESC) as rank_avg_salary
FROM     instructor
GROUP BY college
ORDER BY rank_avg_salary
```

Query 3: rank instructors within colleges by education level with no gaps:

```
SELECT   college, instructor_id, surname, edlevel
DENSERANK() OVER
(PARTITION BY college ORDER BY edlevel DESC) AS rank_edlevel
ORDER BY college, rank_edlevel, surname
```

Query 4: provide row numbers:

```
SELECT   ROWNUMBER() OVER (ORDER BY college, surname) AS number,
surname, annual_salary
FROM     instructor
ORDER BY college, surname
```

Query 5: list the five highest paid instructors:

```
SELECT   instructor_id, surname, annual_salary, rank_salary
FROM     (SELECT instructor_id, surname, annual_salary
         RANK() OVER (ORDER BY annual_salary DESC) AS
         rank_salary
         FROM instructor) AS rank_instructor
WHERE    rank_salary < 6
ORDER BY rank_salary
```

This query uses a nested table expression to compute the rankings before the rank could be used in the WHERE clause. Common table expression could also be used. For more information on table expressions see *Are you recursive*, *DB2 Update*, Issue 101, March 2001.

SUMMARY

OLAP functions provide a valuable new resource for DBAs to satisfy a high percentage of user information analytical requests without resorting to external software or hardware. OLAP functions are useful for DB2 data warehouses and standard DB2 databases.

Eric Garrigue Vesely
Principal/Analyst
Workbench Consulting (Malaysia)

DB2 synchronized dump and CICS reconnect

This article is an implementation of the *DB2 fast close and start* procedure, published in *DB2 Update*, Issues 100 and 101, February and March 2001. The new procedure performs a synchronized dump of DB2 DASD and restarts the DB2/CICS connection. Starting from the previous source code, the new functions are added by a macro source update, that performs the code enhancements. After the DB2 address spaces are ended, the procedure submits and monitors the execution of all DB2 DASD dumps. At the end of dump processing, DB2 is restarted and, if there are CICS sessions to be reconnected to DB2, a DSNC command is issued. The full procedure can be summarized in the following steps:

- Fast DB2 stop subsystem
- DUMP DB2 DASD
- Start DB2
- Restart the CICS/DB2 connection.

PARAMETER DESCRIPTION

The following parameters describe how you can customize the REXX EXEC batch procedure:

- SUBSYS – DB2 subsystem name.
- MAXJOB – number of stop commands executed at the same time (1-8).
- STOSYS – variable for stopping a DB2 subsystem (STOP=YES/ STOP=NO).
- STASYS – variable for starting a DB2 subsystem (START=YES/ START=MAINT/START=NO).
- DMPDASD – to start dumping DB2 DASD (DUMP=YES/ DUMP=NO).

The procedure may be customized for several functions as follows:

- STOP/START USER database:

```
'STOP=no,START=no,DUMP=no'
```

- STOP DB2 and START DB2:

```
'STOP=yes,ARCHLOG=yes,START=yes,DUMP=no'
```

- STOP DB2, DUMP DB2 DASD, and START DB2:

```
'STOP=yes,START=yes,DUMP=yes'
```

CHECKLIST FOR INSTALLATION

Follow these steps to install the components of the REXX procedure:

- Use the preallocated USER.LIBRARY (the user.library allocated for the *DB2 fast close and start* procedure).
- Copy all REXX, macro, parameter, and source enhancements into the USER.LIBRARY in the following members:
  - REXX – @DB2DMP0, @DB2DMP1, @DB2DSNC.
  - Macro – UPDATE00, @MDB2012, @MDB2013, @MDB2037.
  - Parm – DB2DUMP0, DB2DSNC0.
  - Source enhancement: IMPL00, IMPL03, IMPL04, IMPL06, IMPL08.
  - Sample job: STOPDSNZ.
- Run macro UPDATE00 on the original REXX source code @DB2STP0.
- Customize parm DB2DUMP0 with information about the DASD to be dumped, and customize DB2DSNC0 with information about CICS and RCT to be connected to DB2 when it starts.
- Replace macro @MDB2037 and sample job STOPDSNZ.
- Customize job STOPDSNZ according to your environment.
- After installation delete members: IMPL00, IMPL03, IMPL04, IMPL06, IMPL08, and UPDATE00.

The test environment is DB2 Version 5 on OS/390 Version 6.

IMPL00 SOURCE CODE

```
dmpdasd = word(parm,5)
chloin  = word(parm,6)
```

IMPL03 SOURCE CODE

```
 if dmpdasd ¬= 'DUMP=NO' & dmpdasd ¬= 'DUMP=YES' then do
    say '' ; say '' ; say '>>>>>>>>' ; say '>>>>>>>>'
    say '>>>>>>>>  The variable DUMP is wrong !!!!!   '
    say '>>>>>>>>         Specify : DUMP=YES/DUMP=NO  '
    say '>>>>>>>>' ; say '' ; say '' ; exit
    end
 if dmpdasd = 'DUMP=YES' & stosys = 'STOP=NO' then do
  say '' ; say '' ; say '>>>>>>>>'
  say '>>>>>>>> Unable to Start DB2 dump without Stop DB2 subsystem !!!'
  say '>>>>>>>>' ; say '' ; say '' ; exit
    end
```

IMPL04 SOURCE CODE

```
   /*---  DB2 planned Stop   ---*/
 if (today ='Monday' | today ='Wednesday' | today ='Friday') & ,
    ora   < '08:30:00'  & ,
    stosys  = 'STOP=NO' then do
    stosys  = 'STOP=YES'
    stasys  = 'START=YES'
    dmpdasd = 'DUMP=NO'
    chloin  = 'ARCHLOG=NO'
    say '' ; say '>>>>>>>>'
    say '>>>>>>>>  A planned DB2 Stop is running for subsystem 'subsys
    say '>>>>>>>>' ; say ''
    end
   /*---  DB2 planned Dump   ---*/
 if today ='Saturday' & ora < '09:00:00' & stosys = 'STOP=NO' then do
    stosys  = 'STOP=YES'
    stasys  = 'START=YES'
    dmpdasd = 'DUMP=YES'
    chloin  = 'ARCHLOG=NO'
 say '' ; say '>>>>>>>>'
 say '>>>>>>>>  A planned DB2 dump DASD is running for subsystem 'subsys
 say '>>>>>>>>' ; say ''
    end
```

IMPL06 SOURCE CODE

```
   /*---  Dump DB2 DASD  ---*/
    if Dmpdasd = 'DUMP=YES' then do
      say '>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'
```

```
      say '>>>>>>>>  Start DB2 DASD back-up                        >>'
      say '>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'
       say ''
       Call @db2dmpØ subsys
       if word(result,1) = 99 then exit
       end
```

IMPL08 SOURCE CODE

```
   /*---  Open CICS connection   ---*/
       Call @db2dsnc subsys
       if word(result,1) = 99 then exit
```

UPDATE00 MACRO UPDATE

```
/* REXX */
trace ?o
/*-----   Update Macro Enancement   -----*/
say ''
say '          @DB2STPØ REXX update is running.......              '
say ''
isredit macro
isredit find ''implØØ''
call VerRc     implØØ
isredit copy   implØØ after .zcsr
call VerRc     implØØ
call Eximpl    implØØ
isredit change '"if nparm < 4"' '"if nparm < 5"' first
call VerRc     chanØØ
isredit find ''implØ3''
call VerRc     implØ3
isredit copy   implØ3 after .zcsr
call VerRc     implØ3
call Eximpl    implØ3
isredit find ''implØ4''
call VerRc     implØ4
isredit copy   implØ4 after .zcsr
call VerRc     implØ4
call Eximpl    implØ4
isredit find ''implØ6''
call VerRc     implØ6
isredit copy   implØ6 after .zcsr
call VerRc     implØ6
call Eximpl    implØ6
isredit find ''implØ8''
call VerRc     implØ8
isredit copy   implØ8 after .zcsr
call VerRc     implØ8
call Eximpl    implØ8
say ''
```

```
say '    +-------------------------------------------------------+'
say '    |                                                       |'
say '    |      @DB2STPØ REXX update successfully executed       |'
say '    |                                                       |'
say '    +-------------------------------------------------------+'
say ''
isredit end
exit
VerRc :
   arg parmRc
   if rc > Ø then do
say '    +-------------------------------------------------------+  '
say '    |  !!!!!!!!!!!!!!   A T T E N T I O N     !!!!!!!!!!!!!!  |  '
say '    |          Error during Code Update Function            |  '
say '    |            Return Code 'rc' into Step 'parmRc'        |  '
say '    |   Activate trace ?r function and redo the UPDATEØØ Macro |  '
say '    |    on the original Source Code downloaded from Web Site  |  '
say '    +-------------------------------------------------------+  '
      isredit find parmRc
      exit
      end
   return
Exlimpl :
   arg parmRc
   if rc = Ø then do
      isredit exclude parmRc all
      isredit delete x all
      isredit save
      end
   return
```

@DB2DMP0 REXX EXEC

```
/* REXX */
trace ?o
   /*-   +--         Back-up DB2 DASD                --+   */
arg parmin
parm     = translate(parmin,' ',',')
nparm    = words(parm)
subsys   = word(parm,1)
   /*---  Test input parameter   ---*/
 if nparm < 1 then do
    say '' ; say '' ; say '>>>>>>>>'
    say '>>>>>>>>  Parameter list is incomplete !!!!! ' parmin
    say '>>>>>>>>'
    say '' ; say '' ; $exitc = 99
    return $exitc ; exit
    end
   /*---  Parameters assignment   ---*/
 call @db2parØ subsys
 if word(result,1) = 99 then do
```

```
    $exitc = 99 ; return $exitc ; exit
    end
 $lpar    =word(result,1);  $accn    =word(result,2);  $class
=word(result,3)
 $msgcla  =word(result,4);  $region  =word(result,5);  $msglvl
=word(result,6)
 $notif   =word(result,7);  $user    =word(result,8);  $unitda
=word(result,9)
 $unitta  =word(result,10); $esunit  =word(result,11); $prt
=word(result,12)
 $hiwork  =word(result,13); $db2ver  =word(result,14); $ctsubs
=word(result,15)
 $librexx =word(result,16); $parmlib =word(result,17); $proclib
=word(result,18)
 $jcllib  =word(result,19); $report  =word(result,20); $isptenu
=word(result,21)
 $isppenu =word(result,22); $ispmenu =word(result,23); $ispslib
=word(result,24)
 $plilink =word(result,25); $sibmlnk =word(result,26); $sortlib
=word(result,27)
 $runlib  =word(result,28); $dsnload =word(result,29); $tep2pgm
=word(result,30)
 $tep2pln =word(result,31); $unlopgm =word(result,32); $unlopln
=word(result,33)
 $dsnproc =word(result,34)
   /*---   Work areas initialization   ---*/
 blk    =
 idgiul = date(j)
 idor   = space(translate(time(),'',':'),0)
 Dodump = NO
 rit    = S5
 ctrdmp = 0
 #a     = 0
 #d     = 0
 sysid  =
 $user  = BATCH
 $notif = ZZDBADM
   /*---  Read dasd parmlib to be saved   ---*/
 inpds0 =  $parmlib'(db2dump0)'
 xx=outtrap(trp0.)
    "alloc da('"inpds0"') f(dmpdasd) shr reuse"
 xx=outtrap(off)
 if rc > 0 then do
    do a = 1 to trp0.0
       say trp0.a
    end
    say '' ; say ''
    say '>>>>>>>>'
    say '>>>>>>>> "'inpds0'"  Allocation KO '
    say '>>>>>>>>  RC='rc'. Verify.         '
    say '>>>>>>>>'
    say '' ; say ''
```

```
   $exitc = 99 ; return $exitc ; exit
   end
xx=outtrap(trp1.)
   "execio * diskr dmpdasd (stem dmpdasd. finis"
   "free fi(dmpdasd)"
xx=outtrap(off)
if rc > Ø then do
   do a = 1 to trp1.Ø
      say trp1.a
   end
   say '' ; say '' ; say '>>>>>>>>'
   say '>>>>>>>>  Error reading file "'inpdsØ'"'
   say '>>>>>>>>  RC='rc'. Verify.             '
   say '>>>>>>>>'
   say '' ; say '' ; $exitc = 99 ; return $exitc ; exit
   end
  /*---  Dump job allocation file   ---*/
outdsdmp=  $hiwork'.'subsys'.@DB2DMPØ.JOBDUMP'
prmalloc = subsys' 'outdsdmp' Ø 15,15 f,b 8Ø 2792Ø fidmp no'
call @db2allØ prmalloc
if word(result,1) = 99 then do
   $exitc = 99 ; return $exitc ; exit
   end
  /*---  Create dump job   ---*/
say ''
DO #d = 1 to dmpdasd.Ø
   f = Ø  ; s = Ø
   diskdmp.#d = word(dmpdasd.#d,1)
   typedmp.#d = word(dmpdasd.#d,2)
   unitdmp.#d = word(dmpdasd.#d,3)
   okdmp.#d   = word(dmpdasd.#d,4)
   subs.#d    = word(dmpdasd.#d,5)
   if okdmp.#d = Y then do
      if subs.#d = subsys then do
         ctrdmp = ctrdmp  + 1
         Dodump = YES
         select
         when typedmp.#d = 'TPL' then do  /*-- Triple  density --*/
            nop ; end
         when typedmp.#d = 'DBL' then do  /*-- Double  density --*/
            nop ; end
         when typedmp.#d = 'SGL' then do  /*-- singol  density --*/
            nop ; end
         when typedmp.#d = 'LGC' then do  /*-- Logical dump    --*/
            nop ; end
         otherwise
     say '>>>>>>>>'
     say '>>>>>>>>  'typedmp.#d' Unknown type dasd.                  '
     say '>>>>>>>>   No back-up will be produced for volume 'diskdmp.#d
     say '>>>>>>>>'
     say ''
```

```
            ctrdmp = ctrdmp - 1
            iterate
            end
         jobw = FIDMP
         "alloc da('"outdsdmp"') f("jobw") shr reuse"
         outdsver = $hiwork'.'diskdmp.#d'.JOBVER'
      /*---  Dataset name for six months back-up   ---*/
         if substr(date(e),1,2) < 8 then
            rit = M6
      /*---  Physical dump   ---*/
         sysid = substr(subsys,4,1)
         if typedmp.#d = 'TPL' | ,
            typedmp.#d = 'DBL' | ,
            typedmp.#d = 'SGL' then do
            jna = '$'||sysid||diskdmp.#d
sk.1='//'jna' JOB ('$accn'),''DB2-Managment'',CLASS='$class',          '
sk.2='//      MSGCLASS='$msgcla',USER='$user',REGION='$region',       '
sk.3='//      MSGLEVEL=('$msglvl'),NOTIFY='$notif
sk.4='//*     TYPRUN=HOLD                                             '
sk.5='/*JOBPARM BYTES=999999,LINES=9999                               '
sk.6='//* ------------------------------------------------- *        '
sk.7='//*         P h y s i c a l    D u m p   ' diskdmp.#d'        * '
sk.8='//* ------------------------------------------------- *        '
sk.9='//STEP1  EXEC  PGM=ADRDSSU,PARM=''UTILMSG=YES,XABUFF=ABOVE16''  '
sk.1Ø='//SYSPRINT  DD SYSOUT=*                                        '
sk.11='//INPDD1    DD VOL=SER='diskdmp.#d',UNIT='unitdmp.#d',DISP=OLD '
sk.12='//OUTDD1    DD DSN=SYSG.V'diskdmp.#d'.'rit'.D'idgiul'.T'idor', '
sk.13='//          DISP=(,CATLG,DELETE),LABEL=(1,SL),                 '
sk.14='//          VOL=(,RETAIN,,99),UNIT='$unitta
sk.15='//SYSIN     DD  *                                              '
sk.16='  DUMP FULL                -                                   '
sk.17='       INDDNAME (          -                                   '
sk.18='             INPDD1        -                                   '
sk.19='                )          -                                   '
sk.2Ø='       OUTDDNAME(          -                                   '
sk.21='             OUTDD1        -                                   '
sk.22='                )          -                                   '
sk.23='       ALLEXCP             -                                   '
sk.24='       CANCELERROR         -                                   '
sk.25='       COMPRESS            -                                   '
sk.26='       OPTIMIZE(4)         -                                   '
sk.27='       WAIT(Ø,Ø)                                               '
sk.28='//* ------------------------------------------------- *        '
            sk.Ø=28
            Call writerec
            end
      /*---  Logical Dump   ---*/
         if typedmp.#d = 'LGC' then do
            jna = '$'||sysid||substr(diskdmp.#d,1,6)
sk.1='//'jna' JOB ('$accn'),''DB2-Managment'',CLASS='$class',          '
sk.2='//      MSGCLASS='$msgcla',USER='$user',REGION='$region',       '
```

```
sk.3='//      MSGLEVEL=('$msglvl'),NOTIFY='$notif
sk.4='//*     TYPRUN=HOLD                                                '
sk.5='/*JOBPARM BYTES=999999,LINES=9999                                  '
sk.6='//* ------------------------------------------------- *           '
sk.7='//*        L o g i c a l    D u m p     'diskdmp.#d '        *     '
sk.8='//* ------------------------------------------------- *           '
sk.9='//STEP1  EXEC  PGM=ADRDSSU,PARM=''UTILMSG=YES,XABUFF=ABOVE16''     '
sk.1Ø='//SYSPRINT  DD SYSOUT=*                                           '
sk.11='//OUTDD1    DD
DSN=SYSG.V'diskdmp.#d'.'rit'.D'idgiul'.T'idor'.LOGICAL,'
sk.12='//          DISP=(,CATLG,DELETE),LABEL=(1,SL),                    '
sk.13='//          VOL=(,RETAIN,,99),UNIT='$unitta
sk.14='//SYSIN     DD  *                                                 '
sk.15='  DUMP DATASET(             -                                     '
sk.16='       INCLUDE(             -                                     '
sk.17='         'diskdmp.#d'.**    -                                     '
sk.18='              ))            -                                     '
sk.19='       OUTDDNAME(           -                                     '
sk.2Ø='            OUTDD1         -                                     '
sk.21='              )            -                                     '
sk.22='       ALLEXCP              -                                     '
sk.23='       CANCELERROR          -                                     '
sk.24='       SHARE                -                                     '
sk.25='       TOL(ENQF)            -                                     '
sk.26='       COMPRESS             -                                     '
sk.27='       OPTIMIZE(4)          -                                     '
sk.28='       WAIT(Ø,Ø)                                                  '
sk.29='//* ------------------------------------------------- *           '
            sk.Ø=29
            Call writerec
            end
   /*---  Submit job   ---*/
         say '>>>>>>>>  'time() 'Dump 'diskdmp.#d
         address tso "submit '"outdsdmp"'"
         say ''
         end
      end
   end
 say '' ; say '>>>>>>>>'
 say '>>>>>>>>  'ctrdmp' job Dumps has been submitted'
 say '>>>>>>>>' ; say ''
  /*---  Call back-up monitor activity   ---*/
 call @db2dmp1 subsys
 if word(result,1) = 99 then do
   $exitc = 99 ; return $exitc ; exit
   end
 if Dodump = NO then do
 say '' ; say '' ; say '>>>>>>>>'
 say '>>>>>>>>  There are no Dumps to be scheduled for subsystem 'subsys
 say '>>>>>>>>' ; say '' ; say ''
   end
```

```
 xx=outtrap(trpdummy.)
    "free fi(fidmp)"
    address tso "delete '"outdsdmp"'"
 xx=outtrap(off)
 say '' ; say ''
 exit
   /*---  Write routine output records   ---*/
 Writerec :
    "EXECIO * DISKW "jobw" (STEM sk. FINIS"
 Clean:
    DO f = 1 to sk.Ø
       sk.f = blk
       end
    return
```

@DB2DMP1 REXX EXEC

```
/* REXX */
trace ?o
   /*-   +--      DB2 DASD monitor back-up            --+   */
arg parmin
parm    = translate(parmin,' ',',')
nparm   = words(parm)
subsys  = word(parm,1)
   /*---  Test input parameter   ---*/
 if nparm < 1 then do
    say '' ; say '' ; say '>>>>>>>>'
    say '>>>>>>>>  Parameter list is incomplete !!!!! ' parmin
    say '>>>>>>>>'
    say '' ; say '' ; $exitc = 99 ; return $exitc ; exit
    end
   /*---  Parameters assignment   ---*/
 call @db2parØ subsys
 if word(result,1) = 99 then do
    $exitc = 99 ; return $exitc ; exit
    end
 $lpar    =word(result,1);  $accn    =word(result,2);  $class
=word(result,3)
 $msgcla  =word(result,4);  $region  =word(result,5);  $msglvl
=word(result,6)
 $notif   =word(result,7);  $user    =word(result,8);  $unitda
=word(result,9)
 $unitta  =word(result,1Ø); $esunit  =word(result,11); $prt
=word(result,12)
 $hiwork  =word(result,13); $db2ver  =word(result,14); $ctsubs
=word(result,15)
 $librexx =word(result,16); $parmlib =word(result,17); $proclib
=word(result,18)
 $jcllib  =word(result,19); $report  =word(result,2Ø); $isptenu
=word(result,21)
```

```
 $isppenu =word(result,22); $ispmenu =word(result,23); $ispslib
=word(result,24)
 $plilink =word(result,25); $sibmlnk =word(result,26); $sortlib
=word(result,27)
 $runlib  =word(result,28); $dsnload =word(result,29); $tep2pgm
=word(result,3Ø)
 $tep2pln =word(result,31); $unlopgm =word(result,32); $unlopln
=word(result,33)
 $dsnproc =word(result,34)
   /*---   Work areas initialization   ---*/
 ctr     = 1
 ctr1    = Ø
 sysid   = substr(subsys,4,1)
   /*---   Start monitor   ---*/
 say ''
 say '      ********************************************************  '
 say '    *                                              'time()'   *'
 say '    *            Starting monitor for DASD $'sysid'xxxxxx      *'
 say '    *                   subsystem 'subsys'                     *'
 say '    *                                                          *'
 say '    *                          Please Stand By........         *'
 say '    *                                                          *'
 say '      ********************************************************  '
 say ''
   /*---   Verify backup   ---*/
 parmsdsf = subsys || ',$'sysid'*,12Ø,15Ø'
 Call @db2sdsf parmsdsf
 if word(result,1) = 99 then  do
    $exitc = 99 ; return $exitc ; exit
    end
   /*---   Timeout  Exit   ---*/
 if word(result,1) = 98 then  do
    jctr = right(word(result,2),3,'Ø')
 say ''
 say '>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'
 say '>>>>>>>           !!!!!!   W A R N I N G    !!!!!!        >>>>>>>'
 say '>>>>>>>                                                   >>>>>>>'
 say '>>>>>>> 5 hours are spent from the beginning of DUMP DASD >>>>>>>'
 say '>>>>>>> There are still 'jctr' job running.               >>>>>>>'
 say '>>>>>>> Verify the cause of the problem                    >>>>>>>'
 say '>>>>>>> The DB2 subsystem will be started                  >>>>>>>'
 say '>>>>>>>                                                   >>>>>>>'
 say '>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'
    say ''
    $exitc = ØØ
    return $exitc
    exit
    end
 if word(result,1) = ØØ then  do
    say '' ; say '>>>>>>>>'
    say '>>>>>>>>  DB2 DASD back-up jobs are ended'
```



```
    say '>>>>>>>>'
    end
 exit
```

@DB2DSNC REXX EXEC

```
/* REXX */
trace ?o
   /*-   +--          Start CICS connection          --+   */
arg parmin
parm   = translate(parmin,' ',',')
nparm  = words(parm)
Subsys = word(parm,1)
   /*---  Test input parameter   ---*/
 if nparm < 1 then do
    say '' ; say '' ; say '>>>>>>>>'
    say '>>>>>>>>  Parameter list is incomplete !!!!! ' parmin
    say '>>>>>>>>' ; say '' ; say '' ; $exitc = 99 ; return $exitc ;
exit
    end
   /*---  Parameters assignment   ---*/
 Call @db2par0 subsys
 if word(result,1) = 99 then do
    $exitc = 99 ; return $exitc ; exit
    end
 $lpar=word(result,1); $accn=word(result,2);  $class=word(result,3)
 $msgcla=word(result,4);  $region=word(result,5); $msglvl=word(result,6)
 $notif=word(result,7); $user=word(result,8); $unitda=word(result,9)
 $unitta=word(result,10); $esunit=word(result,11); $prt=word(result,12)
 $hiwork=word(result,13); $db2ver=word(result,14); $ctsubs
=word(result,15)
 $librexx =word(result,16); $parmlib =word(result,17); $proclib
=word(result,18)
 $jcllib  =word(result,19); $report  =word(result,20); $isptenu
=word(result,21)
 $isppenu =word(result,22); $ispmenu =word(result,23); $ispslib
=word(result,24)
 $plilink =word(result,25); $sibmlnk =word(result,26); $sortlib
=word(result,27)
 $runlib  =word(result,28); $dsnload =word(result,29); $tep2pgm
=word(result,30)
 $tep2pln =word(result,31); $unlopgm =word(result,32); $unlopln
=word(result,33)
 $dsnproc =word(result,34)
   /*---   Work areas initialization   ---*/
 blk =
 delon = no
 #e  = 0
 #f  = 1
   /*---  Read parmlib for CICS to be connected   ---*/
```

```
inpds0 =  $parmlib'(db2dsnc0)'
xx=outtrap(trp0.)
   "alloc da('"inpds0"') f(db2dsnc) shr reuse"
xx=outtrap(off)
if rc > 0 then do
   do a = 1 to trp0.0
      say trp0.a
   end
   say '' ; say ''
   say '>>>>>>>>'
   say '>>>>>>>> "'inpds0'"  Allocation KO '
   say '>>>>>>>>  RC='rc'. Verify.         '
   say '>>>>>>>>'
   say '' ; say '' ; $exitc = 99 ; return $exitc ; exit
   end
xx=outtrap(trp1.)
   "execio * diskr db2dsnc (stem db2dsnc. finis"
   "free fi(db2dsnc)"
xx=outtrap(off)
if rc > 0 then do
   do a = 1 to trp1.0
      say trp1.a
   end
   say '' ; say '' ; say '>>>>>>>>'
   say '>>>>>>>>  Error reading file "'inpds0'"'
   say '>>>>>>>>  RC='rc'. Verify.             '
   say '>>>>>>>>'
   say '' ; say '' ; $exitc = 99 ; return $exitc ; exit
   end
  /*---  Run DSNC commnd   ---*/
outds0= $hiwork'.'subsys'.@DB2STP0.SYSIN'
jobw = FIOUT
dsn = sysdsn('''outds0''')
if dsn ¬= OK then do
   say ''
   outds0 = $hiwork'.'subsys'.@DB2STP0.SYSIN'
   prmalloc = subsys' 'outds0' 0 5,1 f,b 80 2720 sysin yes'
   call @db2all0 prmalloc
   if word(result,1) = 99 then exit
   delon = yes
   say ''
   end
"alloc da('"outds0"') f("jobw") shr reuse"
DO #d = 1 to db2dsnc.0
   $subsys.#d = word(db2dsnc.#d,1)
   $cix.#d = word(db2dsnc.#d,2)
   $rct.#d = word(db2dsnc.#d,3)
   if $subsys.#d = subsys then do
      #e = #e + 1
      #f = #f + 1
      w0 = "$TA" || substr(subsys,4,1) || right(#f,3,'0')
```

```
       w1 = CA    || substr(subsys,4,1) || right(#f,3,'Ø')
sk.#e="/*"wØ",I=5Ø,'"wØ",''$VS,''''F "$cix.#d",DSNC STRT
"$rct.#d"''''';"w1"''''"
       say '>>>>>>>>  Start CICS attachment facility for '$cix.#d
       end
    end
 if #e > Ø then do
    sk.Ø=#e
    Call writerec
    xx=outtrap(trpØ3.)
       address tso "submit '"outdsØ"'"
    xx=outtrap(off)
    if rc > Ø then do
       do a = 1 to trpØ3.Ø
          say trpØ3.a
          end
       $exitc = 99
       return $exitc
       exit
       end
    end
 else do
 say '>>>>>>>>'
 say '>>>>>>>>  DB2 subsystem has no connection to CICS to be started'
 say '>>>>>>>>'
 say ''
    end
 exit
   /*---  Write routine output records   ---*/
 Writerec :
    "EXECIO * DISKW "jobw" (STEM sk. FINIS"
    DO f = 1 to sk.Ø
       sk.f = blk
       end
    return
```

@MDB2037 EDIT MACRO

```
/* REXX */
trace ?o
/*-----   Macro used in REXX @DB2STPØ      -----*/
isredit macro
isredit exclude all
isredit find ISPSTART
isredit change 'STOP=YES,' 'STOP=NO,' NX
isredit change 'START=YES,' 'START=NO,' NX
isredit change 'START=MAINT,' 'START=NO' NX
isredit change 'DUMP=YES)' 'DUMP=NO)' NX
isredit save
isredit stats off
isredit end
```

SAMPLE BATCH JOB TO SUBMIT DB2 FAST CLOSE PROCEDURE

```
//STOPDSNZ JOB (????0000),'DB2-Management',CLASS=?,MSGCLASS=X,
//       USER=DBA0000,REGION=3M,MSGLEVEL=(1,1),NOTIFY=&SYSUID
/*JOBPARM BYTES=999999,LINES=9999
//* ------------*       Procedure functions          *----------- *
//* ---> STOP/START USER DataBase :                                 *
//*           'STOP=no,START=no,DUMP=no'                            *
//* ---> STOP DB2 + START DB2 :                                     *
//*           'STOP=YES,START=YES,DUMP=no'                          *
//* ---> STOP DB2 + DUMP DB2 DASD + START DB2 :                     *
//*           'STOP=YES,START=YES,DUMP=YES'                         *
//DB2PROC  JCLLIB ORDER=(user.library)
//REXX00   EXEC DB2REXX1
//REXX00.SYSTSIN  DD  *
  ISPSTART CMD(@DB2STP0 DSNZ,8,STOP=NO,START=NO,DUMP=NO)
```

INPUT FOR @DB2DMP0 REXX: VOLUMES TO BE DUMPED

```
DB2DUMP0
+-------+-------+----+-------+----------+
|volume |type of|unit|do/nodo|subsystem |
|       |volume |    |dump   |  DB2     |
+-------+-------+----+-------+----------+
DBZP00    DBL    3390   Y       DSNZ
DBZS00    SGL    3390   Y       DSNZ
DBZS01    TPL    3390   Y       DSNZ
DB2P00    DBL    3390   Y       DSN2
DB2S00    SGL    3390   Y       DSN2
DB2S01    TPL    3390   Y       DSN2
DSNZ001   LGC    XXXX   N       DSNZ
DSN2001   LGC    XXXX   N       DSN2
```

INPUT FOR@DB2DSNC REXX: RCT TO BE CONNECTED

```
DB2DSNC0
+---------+---------+------+
|subsystem|CICS STC |RCT   |
|   DB2   |         |Number|
+---------+---------+------+
  DSNZ     CICSPRF3   0J
  DSNZ     CICSEAM1   07
  DSNE     CICSEAM2   0N
```

Giuseppe Rendano
DB2 Systems Programmer (Italy)

A replacement for DB2/MVS DSNTEP2

DELTEP2 was designed as a replacement for the IBM-supplied DSNTEP2. The problem with DSNTEP2 is that the program is written in PL/I, and provided as source code, not an executable, until you install Version 6. PL/I, as a language, is not supported at many sites (unfortunately). As a result, DSNTEP2 is not easily re-compiled when a new version is made available except by taking the source to a nearby, friendly, PL/I-supporting site and begging (all PL/I sites are friendly by nature)! DB2 Version 6 provides DSNTEP2 as object code that may be bound at the installation.

If you are not aware of DSNTEP2, it is a batch SPUFI, for want of a better definition. SPUFI may not be executed in batch – it requires the ISPF environment; DSNTEP2 fills the hole. IBM does supply DSNTIAD as a sample, but this Assembler program is not very friendly – no comment support, etc, and does not support SELECT statements.

The program DELTEP2 is written in Assembler. Therefore, given some Assembler expertise, it should be more readily maintained. The program is written in 31-bit Assembler, but is resident below 'the line' because DCBs are used for the sequential input and output files. The code is fully re-entrant.

As part of the rewrite, the following 'enhancements' were made to the existing program:

- The program may be executed under the control of the TSO TMP in the usual manner for DB2 batch programs, or the program may be executed to invoke the CAF via DSNALI. If the program is run under the TSO TMP, please execute IKJEFT1B, as opposed to IKJEFT01 – this is nothing new.
- As in DSNTEP2, all current SQL statements (as of Version 5) are supported, including SET CONNECTION *et al.* There are no exceptions – LABEL ON is permitted, despite some statements to the contrary! Just because the statement is supported in the code does not necessarily mean that it may be used; the user is still

required to have the DB2 privileges to perform the action. (Some of these commands are processed as static SQL within DELTEP2 – since the commands may not be processed dynamically.) Remember that CONNECT TO, SET CONNECTION, and RELEASE are not supported by SPUFI – SPUFI is 100% dynamic.

- SQL statements may be prefixed by EXEC SQL (this was also supported in DSNTEP2), and the statement may also be terminated, with no ill effect, by an END-EXEC. (The END-EXEC is *not* followed by a period, but a semi-colon.) This is not supported by DSNTEP2. For some bizarre reason (laziness to be exact), the END-EXEC may be coded without a prior EXEC SQL – this is not a bug, but a feature!
- SPUFI type comments (--) may be included anywhere in the input. This is supported by DSNTEP2. These will be ignored when processing, but will be printed as part of the input statement, and may be used for documentation purposes.
- The old-style DSNTEP2 comment is still supported (an asterisk in column 1). Support is also maintained for the 'standard' DSNTEP2 commands: H – not very informative help – exactly as output by DSNTEP2; HELP – not very informative help – exactly as output by DSNTEP2; EXIT – end of input statements (not required); QUIT – similar to EXIT; END – similar to EXIT.
- These commands are coded in the input stream (horrible PL/I term), just like any SQL command. For example:

```
HELP;
SELECT …;
END;
```

- The input SQL statement is printed out verbatim, line by line, on a separate page from any data being output. This is a change from DSNTEP2. DSNTEP2 'condenses' the statement (removes spaces, comments – the usual unnecessary junk), and prints the first partial page of data, if any, on the same page as the statement. DELTEP2 also condenses the SQL statement, but it is only processed that way, and not printed in a condensed form.

- Run-time parameters have been incorporated, some of which are supported by the latest version of DSNTEP2, but not necessarily the version currently implemented at your site.
  - All parameters are non-positional. This is supported by DSNTEP2.
  - The existing version of DSNTEP2 permits a run-time parameter of ALIGN. The rewrite supports the latest version's implementation: ALIGN(LHS) or ALIGN(MID). This parameter controls, for an SQL SELECT statement, the alignment of the columns in the printed report – LHS indicates Left Hand Side, MID (the default) indicates that the output should be centered.
  - The existing version of DSNTEP2 also supports use of DBCS and SBCS within the same statement. The run-time parameter MIXED or NOMIXED (default) is provided. If MIXED is specified, the Shift-Out (SO) and Shift-In (SI) characters are recognized as the start and end of any DBCS string embedded within a character string. DB2 must be installed with the appropriate capability to allow MIXED format. If a SELECT requests a character string supporting MIXED data, the Shift-Out and Shift-In characters will be printed, if they occur.
- A new run-time parameter, MAXCOL, has been implemented. MAXCOL permits the maximum printed column width for *all* CHAR and VARCHAR columns to be changed, the default being 120. For example, MAXCOL(255) will allow a maximum of 255 characters to be printed for all CHAR and VARCHAR columns. Any excess characters will be ignored – the standard SQLWARN1 flag will be set. This MAXCOL parameter does *not* affect the maximum length of GRAPHIC and VARGRAPHIC columns that may be printed. The default MAXGRAWD is 'equated' to approximately 60 – this is 120 printed bytes. If you want the ability to increase this on demand (not re-assembly), let me know (it will require code changes). When GRAPHIC columns are printed the SO and SI are not shown. If you would like these shown, once again, let me know via e-mail at dkimchi@ibm.net.

- Another new parameter has been incorporated. This is COLHDG(value). This parameter is only meaningful for SELECT statements that reference Table(s) having the LABEL ON applied to one or more columns.

  The values supported are: NAMES (the default) – print the column name (if any) as the heading; LABELS – print the label (or nothing) as the heading; ANY – print the label or, if no label, the column name as the heading.

- SQL and SPUFI permit another value – BOTH. This is not supported in this version of the program. The default value for this parameter is, as stated above, NAMES.

  For selected columns that are unnamed, for example SELECT AVG(col) [AS as_name], the following is printed:

```
with AS    as_name     without AS
(default)  As_name     ?
NAMES      As_name     ?
LABELS     ?           ?
ANY        As_name     ?
```

  For named columns with(out) a LABEL ON:

	Without LABEL ON	*With LABEL ON*	*With AS clause and LABEL ON*
(default)	Column_name	column_name	as_name
NAMES	Column_name	column_name	as_name
LABELS	Column_name	label_name	label_name
ANY	Column_name	label_name	label_name

- To permit the use of CAF, two new, additional parameters are implemented – SSID and PLAN. These permit the specification of the DB2 subsystem name and the PLAN name to be specified respectively. There are no defaults. An example is SSID(DB2T),PLAN(DELTEP2).

- There are a couple of other differences. Printing of single precision floating point numbers is supported. Double precision are converted to single – with the loss of accuracy – and then printed. DB2 does not currently support extended precision (thank goodness). When printing a floating point number, if the value is –1, it is printed as such, not in the standard floating point notation.

Secondly, in pursuance of year 2000 compliance, the date and time are printed as part of the heading on every page.

- There is no difference with regard to the continuation of strings. If a string is not completed before column 73 (by default), just continue immediately on the next input record. Any data coded in 73 is ignored; there is no continuation character; do not continue any variable, other than a string, across multiple input records. Any leading spaces on the second and subsequent statement are treated as part of the character string. When a string is being continued, the subsequent input record(s) will be treated as data, even if marked or coded as comments.

- When a CONNECT TO, a SET CONNECTION, or a RELEASE statement is coded specifying a location, that location may, optionally, be delimited by apostrophes. For example, the following statements are identical:

```
CONNECT TO DB2A;
CONNECT TO $DB2A$;
```

  When a CONNECT or a SET CONNECTION is successful, the message output will provide connection information as follows:

```
CONNECT SUCCESSFUL - PROD: DSN VER: Ø5 REL: Ø1 MOD: Ø
```

  The product, version, release, and modification level of the connected subsystem are shown.

  Neither of these ‘features’ are available within DSNTEP2. In DSNTEP2, the subsystem may not be coded within apostrophes; on a successful CONNECT, no connection information is shown.

- Since a location identifier has, apart from length, the same format as a normal identifier, it may be enclosed in quotes. When enclosed in quotes, apostrophes are permitted as part of the location name. For example, the following are permitted by this program:

```
CONNECT TO DB2A;  -- NORMAL CODING
CONNECT TO 'DB2T';  -- APOSTROPHE DELIMITED
CONNECT TO "HEAVEN'S GATE";  -- NOTE EMBEDDED APOSTROPHE
```

The location name must be specified in the communications database to prevent a –950 SQLCODE from being returned.

SOME EXAMPLES OF USING DELTEP2

Example 1: using call attach

```
//DELTEP2E  EXEC PGM=DELTEP2,
//      PARM='PLAN(DELTEP2),SSID(DB2T),ALIGN(LHS),MAXCOL(255)'
//ABNLDUMP DD SYSOUT=*
//DSNTRACE DD SYSOUT=*
//STEPLIB  DD DISP=SHR,DSN=your.load.library
//         DD DISP=SHR,DSN=DB2T.DSNLOAD
//SYSIN    DD *,DCB=BLKSIZE=8Ø
--
-- GET THE SQL STATEMENTS FOR BPALOG
--
   EXEC SQL          -- OPTIONAL
   SELECT NAME
        , SEQNO
        , TEXT
     FROM SYSIBM.SYSSTMT
    WHERE NAME = 'BPALOG'
      OR TEXT LIKE '%SELECT % FROM % FULL OUTER JOIN % FULL OUTER JOIN %'
 ORDER BY NAME
        , SEQNO
   END-EXEC;         -- FOR COBOLLERS
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
```

Standard SPUFI type comments are shown (two hyphens) within the SQL to be tested.

The ‘LIKE’ predicate is also shown as an implementation of string continuation. I know it is not 72 bytes long, but, if it was, this is how you would do it!

All SQL statements (except the last) must be terminated with a semi-colon.

In this example, the following DD statements are not really required: ABNLDUMP (Abend-Aid) and DSNTRACE (DB2).

Also, do not call me about the DCB=BLKSIZE=80 on the ‘DD *’ statements.

Note the parameters on the EXEC statement. These indicate that the program is attaching to the subsystem DB2T, using a plan name of DELTEP2, the select output will be aligned to the left of the page, and, finally, the maximum column width is to be changed to 255.

The STEPLIB must include the library containing the DELTEP2 load module and the separately loaded DB2 subroutines DSNTIAR and others.

Example 2: using TSO call attach

```
//DELTEP2T  EXEC PGM=IKJEFT1B,DYNAMNBR=20
//STEPLIB  DD DISP=SHR,DSN=your.load.library
//         DD DISP=SHR,DSN=DB2T.DSNLOAD
//SYSIN    DD *,DCB=BLKSIZE=80
   SELECT TEXT
        , NAME
        , SEQNO
     FROM SYSIBM.SYSSTMT
    WHERE NAME = 'BPALOG'
 ORDER BY NAME
        , SEQNO;
//SYSPRINT DD SYSOUT=*
//SYSTSIN  DD *,DCB=BLKSIZE=80
 DSN SYSTEM(DB2T) RETRY(0) TEST(0)
 RUN PROGRAM(DELTEPT)      -
     PLAN(DELTEP2)         -
     LIB('UDQL.LOAD')      -
     PARMS('ALIGN(MID)')
 END
//SYSTSPRT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
```

Once again, do not call me about the DCB=BLKSIZE=80 on the ‘DD *’ statements.

Please note that the RUN, a sub-command of DSN, is continued over multiple lines. The program name here is DELTEPT. Two different load modules are required to give this program the capability to run in either mode, one contains DSNHLI (this one), the other containing DSNELI. Only one plan is required. Very important: notice the continuation indicators ‘–’. Note how well they are aligned; alignment enhances performance!

CLOSING STATEMENTS

This program is supplied as-is with no implied or expressed warranty. I will make attempts to correct any bug, but may have to request machine access to do so. You may freely use this program, if, however, you think it is of some value, make an anonymous contribution to your local cancer society or other medical charitable cause.

Since I am not trusted to have SYSADM or any other useful privilege in most of my work sites, I would like to thank those people who have assisted me in testing this code. This group includes Catalin Comsia, Dave Runnels, Dan Hall, Bill Franklin, and others.

I invite all and sundry to test this DSNTEP2 replacement. Comments (for and against) and recommendations for enhancement are more than welcome.

The best way to reach me is via e-mail at dkimchi@ibm.net.

The areas I would really appreciate someone testing are the DBCS uses. I have tested GRAPHICS (of all varieties) but I have not tested MIXED. The primary installation used for testing does not permit MIXED – this makes testing difficult!

Another problem is printing double and extended precision floating point. Being a lazy person, I would really appreciate it if someone who knows a good, cool, technique (in Assembler – I do not wish to call a COBOL or PL/I subroutine which does some display or put) that they would share it with me. Ideally, this routine should be able to print single, double, and extended precision floating point. I will give credit and be eternally grateful.

If you make any changes to the source code, I am not responsible for those changes, and I would appreciate a copy of the changes and knowing why the changes were required.

The code for this article is available from our Web site at www.xephon.com/extras/deltep.txt.

Dave Loveluck
Consultant (USA)

Quick EXPLAIN for DBRM SQLs

Most of the time, DBAs and application programmers need to run EXPLAIN on a given SQL statement. In a performance tuning exercise, the performance monitors (DB2PM, OMEGAMON, TMON etc) indicate an inefficient SQL statement by its SQL statement number and the DBRM name it belongs to. This REXX EXEC takes the DBRM name and SQL statement number as input, retrieves the corresponding SQL statement text from DBRM, and displays it on the screen. It allows users to make changes, if they wish, to the SQL statement text. Lastly, it runs EXPLAIN for the statement and displays the EXPLAIN results on screen.

When this EXEC is invoked, it displays a panel (panel name PDBRMLI). Users enter the DBRM library name, DBRM member name, DB2 subsystem ID (where EXPLAIN is to be run), tables qualifier ID (this is also the creator ID of the PLAN_TABLE for storing the EXPLAIN results), and the SQL statement number (for the SQL statement on which the EXPLAIN is required). If the entered SQL statement number corresponds to a SQL statement related to OPEN/CLOSE/FETCH CURSOR statements, the program retrieves the statement number for a corresponding DECLARE CURSOR statement from DBRM and displays this SQL statement number *nnn* with a message 'DECLARE CURSOR, GIVE STATEMENT# *nnn*'. This allows the user to re-enter the correct statement number as *nnn* (which belongs to DECLARE CURSOR statement).

After entering these details, the user presses the *Enter* key, and the program extracts the SQL statement text from DBRM and displays the SQL text (in Edit mode) on the screen. Users may like to make changes to the SQL text (they may want to tune the SQL statement by making changes to the text and changing its access path) or may not want to make any changes to the text. Once a user presses PF3, the program reads the edited SQL text, runs EXPLAIN, and displays the following details on the screen (in browse mode):

1 DBRM details.

2 Original SQL statement text (as in the DBRM).

3 Description of the host variables used in the program.

4 Modified SQL text (EXPLAIN is run on this statement).

5 EXPLAIN results.

Once the user has analysed the EXPLAIN results, he can press PF3 and go back to the starting panel to enter the next SQL statement number. To end this EXEC, make the DBRM library and its member name blank and hit the *Enter* key.

Some of the objects used in this REXX EXEC are:

- Load library name for running a dynamic SQL program – XXXXXXX.ssid.RUNLIB.LOAD.
- Dynamic SQL program name – DYNSQL.
- Panel library name – MYLIB.ISPPLIB.
- Panel name – PDBRMLI.

```
/*REXX*/
DUMMY=MSG("OFF")
SSID=''
DYNM_SQL_PGM_LOADLIB='XXXXXXX.ssid.RUNLIB.LOAD'
DYNM_SQL_PGM='DYNSQL'
SQLID=''
HOST_SGN=' : H '
HOST_PRM='   ? '
H_POS=Ø
STMT=''
LIBNAME=''
MEMNAME=''
DBRMOK=''
DBRM=''
MSG=''
EOF='NO'
DARRAY.Ø=Ø
DARRAY. =''
DCTR=Ø
DSQL.Ø=Ø
DSQL. =''
DSQLCTR=Ø
DSTMT.Ø=Ø
DSTMT. =''
DSTMTCTR=Ø
Q1=''
J_IND=Ø
```

```
I=Ø
J=Ø
SQLOK='YES'
ADDRESS "ISPEXEC"
"LIBDEF ISPPLIB DATASET ID ( 'MYLIB.ISPPLIB' )"
DO WHILE EOF='NO'
  MATCH_IND='Ø'
  J_IND=Ø
  DCTR=Ø
  DSTMTCTR=Ø
  DSQLCTR=Ø
  SQLOK='YES'
  "DISPLAY PANEL (PDBRMLI)"
   MSG=''
  IF  ( LIBNAME=' ' ) & ( MEMNAME = ' ' ) THEN EXIT
  LIBNAME=STRIP(LIBNAME)
  MEMNAME=STRIP(MEMNAME)
  DBRM="'"||LIBNAME||'('||MEMNAME||')'||"'"
  DBRMOK=SYSDSN(DBRM)
  IF DBRMOK¬='OK' THEN
      MSG='DBRM LIBRARY OR MEMBER NAME IS INVALID'
  ELSE
      CALL PROCESS_RTN
END /* DO WHILE */
ADDRESS "ISPEXEC"
"LIBDEF ISPPLIB "
EXIT

PROCESS_RTN:
MATCH_IND='Ø'
ADDRESS TSO
CUR_OCF_IND='Ø'
CUR_NAME=''
RECLEN=8Ø
ISTMT=Ø
ISTMT1=Ø
DBRM_STMTNO.Ø=Ø
DBRM_STMTNO. =''
DBRM_STMTTXT.Ø=Ø
DBRM_STMTTXT. =''
CALL INIT
  /* ALLOCATE THE DBRM, READ IT. */
  "ALLOC DD(DBRMIN) DS("DBRM") SHR REUSE"
  "EXECIO * DISKR DBRMIN (STEM LINE. FINIS"
  "FREE DD(DBRMIN)"
  /* CONCATENATE ALL RECORDS INTO ONE LONG LINE.*/
  LONGLINE=''
  DO LOOP=1 TO LINE.Ø
    LONGLINE=LONGLINE||LINE.LOOP
  END /* DO */
```

```
IF LENGTH(LONGLINE)<81 THEN DO
  SAY DBRM 'Is Probably NOT a DBRM...'
  return
END
"NEWSTACK"
/* LET'S BEGIN TO PARSE THE DBRM. MOVE OUTPUT TO A STACK. */
/* DBRM HEADER FIRST. */
/* GET DB2/MVS VER: */
/* IF POS 8Ø=X'ØØ' OR X'C2', IT'S DB2 1.X' */
/* IF POS 8Ø=X'C3', IT'S DB2 2.1 */
/* IF POS 8Ø=X'C4', IT'S DB2 2.2 */
/* IF POS 8Ø=X'C5', IT'S DB2 2.3 */
/* IF POS 8Ø=X'C6', IT'S DB2 3     */
/* IF POS 8Ø=X'C7', IT'S DB2 4.1   */
/* IF POS 8Ø=X'C8', IT'S DB2 5.1   */
/* IF POS 8Ø=X'C9', IT'S DB2 5.1   */
/* IF POS 8Ø=X'D1', IT'S DB2 6.1   */
/* PRIOR TO DB2 2.3 DBRM'S LOOK ABOUT THE SAME. */
DBRM_VER=FALSE
PC_TS_POS=25
PC_TS_LEN=8
VER_INDN_POS=8Ø
VER_INDN_LEN=1
DBRM_HEADER_LEN_POS=5
DBRM_HEADER_LEN_LEN=4
PC_NAME_POS=9
PC_NAME_LEN=8
DBRM_NAME_POS=17
DBRM_NAME_LEN=8
STMTNO_POS=1Ø
SECTNO_POS=8
TKSRCLEN_POS=12
TKSRC_POS=14
DEC1531_PARM_POS=37
QAQQAAAQGN_PARM_POS=33
DATE_TIME_PARM_POS=36
SQL_DIALECT_PARM_POS=38
SQLFLAG_PARM_POS=39
HVDT_POS=2
HVSIZE_POS=4
HVSIZE_DEC_POS_PRE=4
HVSIZE_DEC_POS_LEN=5
HVSIZE_DEC_LEN=1
HVNMLEN_POS=6
HVNM_POS=8
HVHDRLEN=6
DBRMBLOCKLEN_POS=4
VER_INDN=C2X(SUBSTR(LONGLINE,VER_INDN_POS,VER_INDN_LEN))
SELECT
  WHEN VER_INDN='ØØ'|,
```

```
        VER_INDN='C2' THEN DB2_VER='1.X'
    WHEN VER_INDN='C3' THEN DB2_VER='2.1'
    WHEN VER_INDN='C4' THEN DB2_VER='2.2'
    WHEN VER_INDN='C5' THEN DB2_VER='2.3'
    WHEN VER_INDN='C6' THEN DB2_VER='3'
    WHEN VER_INDN='C7' THEN DB2_VER='4.1'
    WHEN VER_INDN='C8' THEN DB2_VER='5.1'
    WHEN VER_INDN='C9' THEN DB2_VER='5.1'
    WHEN VER_INDN='D1' THEN DB2_VER='6.1'
    OTHERWISE
      SAY 'UNSUPPORTED VER OF DB2/MVS'
      return
  END /* SELECT */
  /* AFTER DB2 2.3 DBRMS HAVE A NEW INTERNAL STRUCTURE */
  IF VER_INDN>='C5' THEN DO
    STMTNO_POS=14
    TKSRCLEN_POS=22
    TKSRC_POS=24
    HVDT_POS=17
    HVSIZE_POS=15
    HVSIZE_DEC_POS_PRE=9
    HVSIZE_DEC_POS_LEN=11
    HVSIZE_DEC_LEN=2
    HVSIZE_DEC_POS_PRE_ALT=16
    HVSIZE_DEC_POS_LEN_ALT=15
    HVSIZE_DEC_LEN_ALT=1
    HVNMLEN_POS=22
    HVNM_POS=24
    HVHDRLEN=22
    DBRM_VER=TRUE
    DBRM_VER_LENPOS=81
    DBRM_VER_POS=83
  END
  ADDRESS TSO
  "DELSTACK"
  /* GET THE DBRM NAME & PRECOMP USERID. */
  DBRMNM=SUBSTR(LONGLINE,DBRM_NAME_POS,DBRM_NAME_LEN)
  MAKER=SUBSTR(LONGLINE,PC_NAME_POS,PC_NAME_LEN)
  QUEUE 'DBRM Name: '||DBRMNM
  QUEUE 'Created By: '||MAKER
  QUEUE 'Using DB2 Ver.: '||DB2_VER
  /* GET THE DBRM PRECOMP TIMESTAMP. */
  PC_TIMESTAMP=C2X(SUBSTR(LONGLINE,PC_TS_POS,PC_TS_LEN))
  QUEUE 'Timestamp for Pre-compilation:' PC_TIMESTAMP
  QUEUE ' '
  /* GET THE DBRM VER IF 2.3 OR HIGHER. */
  IF DBRM_VER=TRUE THEN DO
    DBRM_VER_LEN=MAKEDEC(DBRM_VER_LENPOS,2)
    DBRM_VER_STR=SUBSTR(LONGLINE,DBRM_VER_POS,DBRM_VER_LEN)
    IF DBRM_VER_LEN=Ø THEN DBRM_VER_STR='(NONE)'
```

```
  QUEUE 'DBRM Ver. :' DBRM_VER_STR
END
QUEUE 'Parameters Used for Pre-compilation:'
/* GET THE DEC(15|31) PARAMETER. */
DEC1531_PARM=SUBSTR(LONGLINE,DEC1531_PARM_POS,1)
DEC1531='(UNKNOWN)'
IF BITAND(DEC1531_PARM,'00'X)='00'X THEN DEC1531='15'
IF BITAND(DEC1531_PARM,'80'X)='80'X THEN DEC1531='31'
QUEUE '  DEC('DEC1531')'
/* GET THE APOST|QUOTE, APOSTSQL|QUOTESQL, GRAPHIC|NOGRAPHIC PARMS. */
QAQQAAAQGN_PARM=SUBSTR(LONGLINE,QAQQAAAQGN_PARM_POS,1)
GRAPH='(GRAPHIC|NOGRAPHIC UNKNOWN)'
ASQS='(APOSTSQL|QUOTESQL UNKNOWN)'
AQ='(APOST|QUOTE UNKNOWN)'
IF BITAND(QAQQAAAQGN_PARM,'10'X)='00'X THEN GRAPH='NOGRAPHIC'
IF BITAND(QAQQAAAQGN_PARM,'10'X)='10'X THEN GRAPH='GRAPHIC'
IF BITAND(QAQQAAAQGN_PARM,'20'X)='00'X THEN ASQS='APOSTSQL'
IF BITAND(QAQQAAAQGN_PARM,'20'X)='20'X THEN ASQS='QUOTESQL'
IF BITAND(QAQQAAAQGN_PARM,'80'X)='00'X THEN AQ='APOST'
IF BITAND(QAQQAAAQGN_PARM,'80'X)='80'X THEN AQ='QUOTE'
QUEUE '  'GRAPH
QUEUE '  'ASQS
QUEUE '  'AQ
/* GET THE DATE PARAMETER. */
DATE_TIME_PARM=SUBSTR(LONGLINE,DATE_TIME_PARM_POS,1)
DATE_FORMAT='(UNKNOWN)'
TIME_FORMAT='(UNKNOWN)'
IF BITAND(DATE_TIME_PARM,'00'X)='00'X THEN DATE_FORMAT='ISO'
IF BITAND(DATE_TIME_PARM,'01'X)='01'X THEN DATE_FORMAT='USA'
IF BITAND(DATE_TIME_PARM,'02'X)='02'X THEN DATE_FORMAT='JIS'
IF BITAND(DATE_TIME_PARM,'04'X)='04'X THEN DATE_FORMAT='EUR'
IF BITAND(DATE_TIME_PARM,'08'X)='08'X THEN DATE_FORMAT='LOCAL'
IF BITAND(DATE_TIME_PARM,'00'X)='00'X THEN TIME_FORMAT='ISO'
IF BITAND(DATE_TIME_PARM,'10'X)='10'X THEN TIME_FORMAT='USA'
IF BITAND(DATE_TIME_PARM,'20'X)='20'X THEN TIME_FORMAT='JIS'
IF BITAND(DATE_TIME_PARM,'40'X)='40'X THEN TIME_FORMAT='EUR'
IF BITAND(DATE_TIME_PARM,'80'X)='80'X THEN TIME_FORMAT='LOCAL'
QUEUE '  DATE('DATE_FORMAT')'
QUEUE '  TIME('TIME_FORMAT')'
/* GET THE SQL_DIALECT PARAMETER IF VER 2.3 OR UP. */
IF VER_INDN>='C5' THEN DO
  SQL_DIALECT_PARM=SUBSTR(LONGLINE,SQL_DIALECT_PARM_POS,1)
  SQL_DIALECT='(UNKNOWN)'
  IF BITAND(SQL_DIALECT_PARM,'01'X)='01'X THEN SQL_DIALECT='DB2'
  IF BITAND(SQL_DIALECT_PARM,'02'X)='02'X THEN SQL_DIALECT='ALL'
  QUEUE '  SQL('SQL_DIALECT')'
END /* IF */
/* GET THE SQLFLAG PARAMETER. */
SQLFLAG_PARM=SUBSTR(LONGLINE,SQLFLAG_PARM_POS,1)
SQLFLAG='(UNKNOWN)'
```

```
IF BITAND(SQLFLAG_PARM,'ØØ'X)='ØØ'X THEN SQLFLAG='86(SSID,QUALIFIER)' Ø
 IF BITAND(SQLFLAG_PARM,'Ø1'X)='Ø1'X THEN SQLFLAG='SAA'
 QUEUE '  SQLFLAG('SQLFLAG')'
 /* QUIT HEADER PROCESSING. */
 QUEUE ' '
 /* GET EYECATCHER 'DBRM', START LOOKING RIGHT AFTER HEADER; */
 /* THIS STRING INDICATES THE START OF A SQL STATEMENT BLOCK. */
 /* IF NOT FOUND, DBRMPOS IS SET TO ZERO, MEANING THERE ARE */
 /* NO SQL STATEMENT BLOCKS TO PROCESS. */
 JUMP=MAKEDEC(DBRM_HEADER_LEN_POS,DBRM_HEADER_LEN_LEN)
 DBRMPOS=POS('DBRM',LONGLINE,JUMP)
 /* LOOP AS LONG AS THERE AS SQL STATEMENTBLOCKS TO PROCESS. */
 DO WHILE DBRMPOS¬=Ø
   /* GET THE STATEMENT NUMBER (I.E. SYSIBM.SYSSTMT.STMTNO). */
   STMTNO=MAKEDEC(DBRMPOS+STMTNO_POS,2)
   SECTNO=MAKEDEC(DBRMPOS+SECTNO_POS,2)
   SSTATPOS=DBRMPOS+TKSRC_POS
   TKSRCLEN=MAKEDEC(DBRMPOS+TKSRCLEN_POS,2)
   ESTATPOS=SSTATPOS+TKSRCLEN
   IF STMTNO = STMT THEN
     DO
      MATCH_IND='1'
      QUEUE COPIES('*',RECLEN)
    QUEUE '**  Statement Number '||STMTNO||' of Section '||SECTNO||' *'
      QUEUE COPIES('*',RECLEN)
      QUEUE ' '
     END
   STOP=FALSE
   STR=''
   STR_IND='F'
   SQL_TXT=''
   DO WHILE STOP=FALSE
     STRPTR=SSTATPOS+RECLEN
     IF STRPTR>=ESTATPOS THEN DO
       STRPTR=ESTATPOS
       STOP=TRUE
     END
     IF STMTNO = STMT THEN
        DO
           QUEUE SUBSTR(LONGLINE,SSTATPOS,STRPTR-SSTATPOS)
           DSQLCTR=DSQLCTR+1
           DSQL.DSQLCTR=SUBSTR(LONGLINE,SSTATPOS,STRPTR-SSTATPOS)
        END
     SQL_TXT=SQL_TXT||SUBSTR(LONGLINE,SSTATPOS,STRPTR-SSTATPOS)
     SSTATPOS=STRPTR
   END /* DO WHILE STOP=FALSE */
   CUR_OP=WORD(SQL_TXT,1)
   IF ( CUR_OP='OPEN' | CUR_OP='CLOSE' | CUR_OP='FETCH' ) THEN
      IF STMTNO = STMT THEN
         DO
```

```
          CUR_OCF_IND='1'
          CUR_NAME=WORD(SQL_TXT,2)
       END
IF CUR_OP='DECLARE' THEN
   DO
     ISTMT=ISTMT+1
     DBRM_STMTNO.ISTMT=STMTNO
     DBRM_STMTTXT.ISTMT =WORD(SQL_TXT,2)
   END
OFFSET=ESTATPOS
NOFHV=MAKEDEC(OFFSET,2)
OFFSET=OFFSET+2
HVNMLEN=Ø
/* LOOP TO MAKE A REPORT RECORD FOR EVERY HOST VARIABLE. */
IF STMTNO = STMT THEN
   QUEUE ' '
IF STMTNO = STMT THEN
   IF NOFHV>Ø THEN DO
      QUEUE ' '
      QUEUE ' '
      QUEUE 'Host Variables USED in this Statement  '
      QUEUE '_____________________________________ '
      QUEUE ' '
      QUEUE JUSTIFY('Variable_Name',48)||'Length DataType'
      QUEUE COPIES('-',47)||' '||COPIES('-',6)||' '||COPIES('-',25)
   END
DO HVLOOP=1 TO NOFHV
  HVDT=MAKEDEC(OFFSET+HVDT_POS,2)
  HVSIZE=MAKEDEC(OFFSET+HVSIZE_POS,2)
  REM=''
  IF HVDT=484 | HVDT=485 | HVDT=5Ø4 | HVDT=5Ø5 THEN DO
    HVSIZE_PRE=MAKEDEC(OFFSET+HVSIZE_DEC_POS_PRE,HVSIZE_DEC_LEN)
    HVSIZE_LEN=MAKEDEC(OFFSET+HVSIZE_DEC_POS_LEN,HVSIZE_DEC_LEN)
    IF (VER_INDN >='C5') & (HVSIZE_PRE+HVSIZE_LEN=Ø) THEN DO
      HVSIZE_PRE=MAKEDEC(OFFSET+HVSIZE_DEC_POS_PRE_ALT,HVSIZE_DEC_LE
 HVSIZE_LEN=MAKEDEC(OFFSET+HVSIZE_DEC_POS_LEN_ALT,HVSIZE_DEC_LEØ3Ø6ØØ
    END
    HVSIZE=HVSIZE_PRE||','||HVSIZE_LEN
  END /* IF */
  IF HVSIZE=Ø THEN DO
    HVSIZE='-'
    REM='PROBABLY A NULL INDN'
  END /* IF */
  /* IF THE DATATYPE IS 32Ø OR 321, IT'S NOT AN ATTRIBUTE, BUT */
  /* THE NAME OF A HOSTSTRUCTURE. */
  IF HVDT=32Ø | HVDT=321 THEN DO
    HVSIZE='-'
    REM='A HOST LANGUAGE STRUCTURE, IT EXPANDS AS FOLLOWS:'
  END /* IF */
  HVNMLEN=MAKEDEC(OFFSET+HVNMLEN_POS,2)
```

```
      HVNAAM=SUBSTR(LONGLINE,OFFSET+HVNM_POS,HVNMLEN)
      IF STMTNO = STMT THEN
         QUEUE JUSTIFY(HVNAAM,47) JUSTIFY(HVSIZE,5) DATATYPE.HVDT
      IF REM>'' THEN DO
         IF STMTNO = STMT THEN
           QUEUE '  >THE VARIABLE ABOVE IS '||REM
      END
      OFFSET=OFFSET+HVNMLEN+HVHDRLEN
    END /* DO HVLOOP=1 TO NOFHV */
    IF STMTNO = STMT THEN
       DO
         QUEUE COPIES('-',47)||' '||COPIES('-',6)||' '||COPIES('-',25)
         QUEUE ' '
       END
    STATLEN=MAKEDEC(DBRMPOS+DBRMBLOCKLEN_POS,4)
    JUMP=DBRMPOS+STATLEN
    DBRMPOS=POS('DBRM',LONGLINE,JUMP)
  END /* WHILE DBRMPOS¬=Ø */
  IF MATCH_IND<>'1'  THEN
     DO
        MSG="STATEMENT NUMBER IS NOT CORRECT"
        ADDRESS TSO
        "DELSTACK"
        RETURN
     END
  IF CUR_OCF_IND='1'  THEN
     DO ISTMT1 = 1 TO ISTMT
         IF DBRM_STMTTXT.ISTMT1=CUR_NAME  THEN
            DO
             MSG="DECLARE CURSOR, GIVE STATEMENT# "||DBRM_STMTNO.ISTMT1
             ADDRESS TSO
             "DELSTACK"
             "NEWSTACK"
             RETURN
            END
     END
  CALL  EXTRACT_STMT
RETURN
/********/
/* SET UP A TABLE OF SQL DATATYPES, RELATE THEM TO CODES USED BY DB2 */
/* AS DESCRIBED IN THE DB2 APPLICATION PROGRAMMING AND SQL GUIDE. */
INIT:
  DATATYPE.=' (UNKNOWN)                '
  DATATYPE.32Ø=' -                        '
  DATATYPE.321=' -                        '
  DATATYPE.384=' DATE NOT NULL            '
  DATATYPE.385=' DATE                     '
  DATATYPE.388=' TIME NOT NULL            '
  DATATYPE.389=' TIME                     '
  DATATYPE.392=' TIMESTAMP NOT NULL       '
```

```
  DATATYPE.393=' TIMESTAMP                '
  DATATYPE.448=' VARCHAR NOT NULL         '
  DATATYPE.449=' VARCHAR                  '
  DATATYPE.452=' CHAR NOT NULL            '
  DATATYPE.453=' CHAR                     '
  DATATYPE.456=' LONG VARCHAR NOT NULL    '
  DATATYPE.457=' LONG VARCHAR             '
  DATATYPE.460=' NULL TERM CHARSTRING NN  '
  DATATYPE.461=' NULL TERM CHARSTRING     '
  DATATYPE.464=' VARGRAPHIC NOT NULL      '
  DATATYPE.465=' VARGRAPHIC               '
  DATATYPE.468=' GRAPHIC NOT NULL         '
  DATATYPE.469=' GRAPHIC                  '
  DATATYPE.472=' LONG VARGRAPHIC NOT NULL '
  DATATYPE.473=' LONG VARGRAPHIC          '
  DATATYPE.480=' FLOAT NOT NULL           '
  DATATYPE.481=' FLOAT                    '
  DATATYPE.484=' DECIMAL NOT NULL         '
  DATATYPE.485=' DECIMAL                  '
  DATATYPE.496=' INTEGER NOT NULL         '
  DATATYPE.497=' INTEGER                  '
  DATATYPE.500=' SMALLINT NOT NULL        '
  DATATYPE.501=' SMALLINT                 '
  DATATYPE.504=' DECIMAL DISP.SIGN LS NN  '
  DATATYPE.505=' DECIMAL DISP.SIGN LS     '
RETURN
/* FUNCTION MAKEDEC, USED TO MAKE A DECIMAL FROM A CHARACTER STRING. */
MAKEDEC:
  ARG SPOS, LEN
  MAKEDEC=C2D(SUBSTR(LONGLINE,SPOS,LEN))
RETURN MAKEDEC
EXTRACT_STMT:
  ADDRESS TSO
  "ALLOC DD(DDETL) NEW DELETE UNIT(VIO) RECFM (F B) LRECL (80)"
  IF RC <> 0 THEN DO
     SAY 'ERROR IN DDETL ALLOC *' || RC
     RETURN
  END
  QUEUE ''
  "EXECIO * DISKW DDETL (FINIS"
  IF RC <> 0 THEN DO
     SAY     '* ERROR WRITING DDETL FILE *RC* ' || RC
     RETURN
  END
  ADDRESS TSO
  "EXECIO * DISKR DDETL  (FINIS"
  IF RC <> 0 THEN DO
    SAY     '* ERROR READING FILE DDETL   *RC* ' || RC
    RETURN
  END
```

```
  NREC = QUEUED()
  DO J= 1 TO NREC
    PULL RECORD
    DCTR=DCTR+1
    DARRAY.DCTR=RECORD
  END
  ADDRESS TSO
  "DELSTACK"
  "FREE  FI(DDETL)"
  "ALLOC DD(SYSPRINT) NEW DELETE UNIT(VIO)"
  IF RC <> Ø THEN DO
    SAY 'ERROR IN SYSPRINT ALLOC *' || RC
    RETURN
  END
  CALL BUILD_SYSIN
  ADDRESS TSO
  "DELSTACK"
  CALL PØØØ1_BLD_THDLIST
  ADDRESS TSO
  "FREE  FI(SYSIN)"
  "FREE  FI(SYSPRINT)"
RETURN
/*****************************************************************/
/* PØØØ1_BLD_THDLIST                                             */
/*****************************************************************/
PØØØ1_BLD_THDLIST:
    DUMMY = OUTTRAP("OUTPUT_LINE.","*")
    Q1=" RUN PROGRAM ("||DYNM_SQL_PGM||") PLAN("||DYNM_SQL_PGM||") "
    Q1=Q1||"  LIB ('"||DYNM_SQL_PGM_LOADLIB||"')"
    QUEUE Q1
    QUEUE "END"
    ADDRESS TSO
    "DSN SYSTEM("||SSID||")"
    IF RC > 4 THEN
      DO
        SQLOK='NO'
        SAY 'DSN COMMAND  FAILED,  RETURN CODE = '||RC
      END
    CALL BUILT_REPORT
    CALL BROWSE_REPORT
RETURN
BUILT_REPORT:
  DCTR=DCTR+1
  DARRAY.DCTR='                                                '
  DCTR=DCTR+1
  DARRAY.DCTR='                                                '
  DCTR=DCTR+1
  DARRAY.DCTR='EXPLAIN RESULTS for SQL Statement'
  DCTR=DCTR+1
  DARRAY.DCTR='_____________________________________           '
```

```
  DCTR=DCTR+1
  DARRAY.DCTR='                                                          '
  ADDRESS TSO
  "EXECIO * DISKR SYSPRINT (FINIS"
  IF RC ¬= Ø THEN DO
    SAY     '* ERROR READING INPUT FILE   *RC* ' || RC
    RETURN
  END
  NREC = QUEUED()
  DO J= 1 TO NREC
    PULL RECORD
    IF J_IND=Ø & SQLOK='YES' THEN
      IF SUBSTR(RECORD,1Ø,1Ø) <> '+---------' THEN
        ITERATE
      ELSE
        DO
           J_IND=1
           DCTR=DCTR+1
           DARRAY.DCTR=RECORD
        END
    ELSE
      DO
         IF SUBSTR(RECORD,1,6) =  '1PAGE ' THEN
            ITERATE
         IF SUBSTR(RECORD,2,2Ø) =  'SUCCESSFUL RETRIEVAL' THEN
            ITERATE
         DCTR=DCTR+1
         DARRAY.DCTR=RECORD
      END
  END
  ADDRESS TSO
  "DELSTACK"
RETURN
BROWSE_REPORT:
    ADDRESS TSO
    "DELSTACK"
    ADDRESS TSO
    "NEWSTACK"
    "ALLOC DD(DDDESCR) NEW DELETE UNIT(VIO) RECFM (F B) LRECL (133)"
    IF RC ¬= Ø THEN DO
       SAY 'ERROR IN DDDESCR ALLOC *' || RC
       RETURN
    END
    DO J= 1 TO DCTR
      RECORD = DARRAY.J
      QUEUE RECORD
    END
    QUEUE ''
    "EXECIO * DISKW DDDESCR (FINIS"
    IF RC ¬= Ø THEN DO
```

```
       SAY     '* ERROR WRITING DDDESCR FILE *RC* ' || RC
       RETURN
    END
    ADDRESS ISPEXEC
    "LMINIT DATAID(DDVAR)  DDNAME(DDDESCR) "
    "BROWSE DATAID(" DDVAR ")"
    "LMFREE DATAID(" DDVAR ")"
    ADDRESS TSO
    "FREE  FI(DDDESCR)"
RETURN
BUILD_SYSIN:
    ADDRESS TSO
    "DELSTACK"
    CALL EDIT_STMT
    CALL UPDT_STMT
    ADDRESS TSO
    "DELSTACK"
    "ALLOC DD(SYSIN) NEW DELETE UNIT(VIO) RECFM (F B) LRECL (8Ø)"
    IF RC ¬= Ø THEN DO
       SAY 'ERROR IN SYSIN ALLOC *' || RC
       RETURN
    END
    CALL BUILD_DEL
    CALL BUILD_STMT
    CALL BUILD_EXPL
    QUEUE ''
    "EXECIO * DISKW SYSIN (FINIS"
    IF RC ¬= Ø THEN DO
       SAY     '* ERROR WRITING SYSIN FILE *RC* ' || RC
       RETURN
    END
RETURN
BUILD_DEL:
  RECORD  = " SET CURRENT SQLID = '"||SQLID||"' ; "
  QUEUE RECORD
  RECORD= "DELETE FROM   PLAN_TABLE "
  QUEUE RECORD
  RECORD= "WHERE  QUERYNO  =    4Ø   ; "
  QUEUE RECORD
  RECORD  = "EXPLAIN PLAN SET QUERYNO = 4Ø FOR  "
  QUEUE RECORD
RETURN
BUILD_EXPL:
   RECORD = " SELECT *  FROM PLAN_TABLE  "
   QUEUE RECORD
   RECORD = " WHERE QUERYNO = 4Ø  "
   QUEUE RECORD
   RECORD = " ORDER BY APPLNAME, PROGNAME, QUERYNO, QBLOCKNO, PLANNO;"
   QUEUE RECORD
RETURN
```

```
EDIT_STMT:
 REC1=''
 REC2=''
 LOOP_STOP='NO'
 H_POS=Ø
 DO J = 1 TO DSQLCTR
    REC1=REC1||DSQL.J
 END
 DO WHILE LOOP_STOP='NO'
    H_POS=POS(HOST_SGN,REC1)
    IF H_POS=Ø THEN
       DO
          REC2=REC2||REC1
          LOOP_STOP='YES'
       END
    ELSE
       DO
          REC2=REC2||SUBSTR(REC1,1,H_POS)||HOST_PRM
          REC1=SUBSTR(REC1,H_POS+4)
       END
 END
 REC2=SPACE(REC2)
 H_POS=POS(' INTO ?',REC2)
 IF (H_POS > Ø & WORD(rec2,1)='SELECT')   THEN
    DO
       REC1=''
       REC1=REC1||SUBSTR(REC2,1,H_POS)
       REC2=SUBSTR(REC2,H_POS+7)
       H_POS=POS(' FROM ',REC2)
       IF H_POS > Ø THEN
          REC1=REC1||SUBSTR(REC2,H_POS)
    END
 ELSE
    REC1=REC2
 H_POS=Ø
 REC1=SPACE(REC1)
 IF WORD(REC1,1)='DECLARE' THEN
    DO
       REC2=''
       H_POS=POS(' SELECT ',REC1)
       IF H_POS > Ø THEN
          REC2=SUBSTR(REC1,H_POS)
    END
 ELSE
    REC2=REC1
 REC1=SPACE(REC2)
 H_POS=Ø
 H_POS=POS(' WHERE CURRENT OF ',REC1)
 IF H_POS > Ø THEN
    REC2=SUBSTR(REC1,1,H_POS)||' '
```

```
ELSE
   REC2=REC1
REC1=SPACE(REC2)
LOOP_STOP='NO'
H_POS=Ø
REC4=''
REC3=''
REC2=''
NWORDS=Ø
REC1_LEN=Ø
DO WHILE LOOP_STOP='NO'
   REC1_LEN=LENGTH(REC1)
   IF REC1_LEN >  7Ø  THEN
      DO
         REC4=REC1
         REC3=SUBSTR(REC1,1,7Ø)
         REC1=SUBSTR(REC1,71)
         IF (SUBSTR(REC4,7Ø,1) = ' '  |  SUBSTR(REC4,71,1)=' ') THEN
           DO
              RECORD = REC3
              QUEUE RECORD
           END
         ELSE
           DO
               NWORDS=WORDS(REC3)
               IF NWORDS > 1 THEN
                  DO
                     REC2=SUBWORD(REC3,1,NWORDS-1)||' '
                     RECORD = REC2
                     QUEUE RECORD
                     REC2=' '||SUBWORD(REC3,NWORDS)
                  END
               ELSE
                  REC2=' '||SUBWORD(REC3,1)
               REC2=REC2||SUBWORD(REC1,1,1)||' '
               RECORD = REC2
               QUEUE RECORD
               REC1=' '||SUBWORD(REC1,2)
           END
      END
   ELSE
      DO
         RECORD = REC1
         QUEUE RECORD
         REC1=''
         LOOP_STOP='YES'
      END
  IF WORDS(REC1) = Ø THEN
     LOOP_STOP='YES'
END
```

```
 QUEUE ' ; '
RETURN
UPDT_STMT:
  ADDRESS TSO
  "ALLOC DD(STMTF) NEW DELETE UNIT(VIO) RECFM (F B) LRECL (8Ø)"
  IF RC <> Ø THEN DO
     SAY 'ERROR IN STMTF ALLOC *' || RC
     RETURN
  END
  QUEUE ''
  "EXECIO * DISKW STMTF (FINIS"
  IF RC <> Ø THEN DO
     SAY     '* ERROR WRITING STMTF FILE *RC* ' || RC
     RETURN
  END
  ADDRESS ISPEXEC
  "LMINIT DATAID(DDVAR1)  DDNAME(STMTF) "
  "EDIT   DATAID(" DDVAR1 ")"
  "LMFREE DATAID(" DDVAR1 ")"
  ADDRESS TSO
  "EXECIO * DISKR STMTF  (FINIS"
  IF RC <> Ø THEN DO
    SAY     '* ERROR READING FILE STMTF   *RC* ' || RC
    RETURN
  END
  NREC1 = QUEUED()
  DO J= 1 TO NREC1
    PULL RECORD
    DSTMTCTR=DSTMTCTR+1
    DSTMT.DSTMTCTR=RECORD
  END
  ADDRESS TSO
  "DELSTACK"
  "FREE  FI(STMTF)"
RETURN
BUILD_STMT:
  DCTR=DCTR+1
  DARRAY.DCTR='                                                    '
  DCTR=DCTR+1
  DARRAY.DCTR='                                                    '
  DCTR=DCTR+1
  DARRAY.DCTR='SQL Statement - to be Explained'
  DCTR=DCTR+1
  DARRAY.DCTR='_____________________________________               '
  DCTR=DCTR+1
  DARRAY.DCTR='                                                    '
  DO J= 1 TO DSTMTCTR
    RECORD=DSTMT.J
    QUEUE RECORD
    DCTR=DCTR+1
```

```
    DARRAY.DCTR=RECORD
  END
RETURN
```

PANEL PDBRMLI

```
)ATTR
/*****************************************************************************/
/* Pdbrmli - display dbrmlib                                                 */
/*****************************************************************************/
   + TYPE(TEXT)   INTENS(LOW)   COLOR(BLUE)         SKIP(ON)
   % TYPE(TEXT)   INTENS(HIGH)  COLOR(WHITE)        SKIP(ON)
   $ TYPE(TEXT)   INTENS(HIGH)  COLOR(RED)          SKIP(ON)
   # TYPE(OUTPUT) INTENS(HIGH)  COLOR(TURQUOISE)    CAPS(ON)
)BODY CMD(C)
%---------------------+ QUICK EXPLAIN FOR DBRMS  %--------------------
%OPTION ===_C                                              %SCR-_AMT +
+                                                                      +
+                                                                      +
+                                                                      +
+    %Enter LIBRARY & MEMBER names        or Blanks to  Quit +         +
+                                                                      +
+                                                                      +
+                                                                      +
+    %DBRM Library Name:_z                                       +     +
+    %DBRM member  Name:_z                                       +     +
+    %Sub-System   Name:_z                                       +     +
+    %Tables Qualifier :_z                                       +     +
+    %Statement  Number:_z                                       +     +
+                                                                      +
+
+                                                                      +
+
+                                                                      +
+ #msg                                                            +
+                                                                      +
+
)INIT
 .ZVARS = '(libname,memname,ssid,sqlid,stmt)'
)END
```

Sharad K Pande
Senior DB2 DBA
PriceWaterhouseCoopers (USA)

Bunker Hill has began shipping its ei-O database development and management software, which automates the migration of Microsoft Access applications to IBM mainframes running DB2 on OS/390 and z/OS. It also helps to simplify on-going DB2 maintenance, new development, and database administration.

The software enables Web-enabled server consolidation. In addition to database migration, ei-O (electronic input-Output) adapts the Access front-end application to operate using the migrated DB2 data. The adapted Access database file can be saved centrally and opened using a browser. It eliminates the scalability, network performance, and disaster recovery shortcomings of Access, and helps regain control of data and program code while preserving investments in Access user interfaces and training.

Besides migration capabilities, it also enables programmers and others to prototype, build, and synchronize DB2 databases using the Access rapid development environment.

For further information contact:
Bunker Hill , 501 Delancey Street, Suite 609, San Francisco, CA 94107, USA.
Tel: (925) 328 1103.
URL: http://www.bunkerhill.com/ei-Onews.htm.

* * *

Open Market Inc has announced that its content management products will offer native support for IBM's Websphere Application Server and DB2 UDB.

Open Market's Content Server and associated applications, Content Centre, Personalization Centre, Catalog Centre, and Marketing Studio, offer native support for a variety of J2EE-compliant application servers.

For further information contact:
Open Market Inc, One Wayside Road, Burlington, MA, 01803, USA.
Tel: (781) 359 3000.
URL: http://www.openmarket.com.

* * *

Alysis Technologies has announced that it will bundle Workout, its electronic billing software, with IBM e-business solutions DB2 UDB and Websphere Application Server.

Workout is Internet billing software that is designed to overcome service and security issues.

For further information contact:
Alysis Technologies, 1900 Powell Street, Suite 500 Emeryville, CA 94608-1840, USA.
Tel: (510) 450 7000.

Alysis Technologies, Arundel House, 31 A St James's Square, London SW1Y 4JR, UK.
Tel: (020) 7665 1645.
URL: http://www.alysis.com/products/workoutb2b.html.