139

DB2 update

May 2004

In this issue

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Nicole Thomas
E-mail: nicole@xephon.com

Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs $380.00 in the USA and Canada; £255.00 in the UK; £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 2000 issue, are available separately to subscribers for $33.75 (£22.50) each including postage.

DB2 Update on-line

Code from *DB2 Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/db2; you will need to supply a word from the printed issue.

Contributions

When Xephon is given copyright, articles published in *DB2 Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

DB2 UDB V8 LUW – non-index data retrieval

This article looks at a way of retrieving non-key data by using the index and thereby avoiding going to the underlying data pages.

There is a feature in DB2 UDB for LUW that allows you effectively to store data in the index. This extra data is not used to enforce uniqueness, but having it in the index means that you do not have to go to the underlying data pages to get it – which should be better from a performance point of view.

I ran all the SQL in this article on a Windows 2000 machine running DB2 UDB 8.1 FP2.

Let's use the EMPLOYEE table in the SAMPLE database. The table is made up of the following columns: empno, firstnme, midinit, lastname, workdept, phoneno, hiredate, job, edlevel, sex, birthdate, salary, bonus, and comm.

I wanted to select the EMPNO and the PHONENO from the table. I performed two tests. In the first test I created an index on EMPNO and selected from the table, and in the second test I created an index on EMPNO specifying PHONENO as a non-key part of the index. I used **db2expln** to examine the access path and costs using both indexes.

So for the first test, I created an index on empno:

```
>db2 connect to sample

>db2 create unique index emp_ix1 on employee (empno)
```

Now, running the SQL through **db2expln**:

```
>db2expln -d sample -q "select phoneno from employee where empno =
'000010'"  -t
```

and writing the output to the terminal, gives:

```
SQL Statement:

  select phoneno
```

```
  from employee
  where empno = '000010'

Section Code Page = 1252

Estimated Cost = 25.045372
Estimated Cardinality = 1.000000

Access Table Name = DB2ADMIN.EMPLOYEE  ID = 2,5
|  #Columns = 2
|  Single Record
|  Index Scan:  Name = DB2ADMIN.EMP_IX1  ID = 1
|  |  Regular Index (Not Clustered)
|  |  Index Columns:
|  |  |  1: EMPNO (Ascending)
|  |  #Key Columns = 1
|  |  |  Start Key: Inclusive Value
|  |  |  |  1: '000010'
|  |  |  Stop Key: Inclusive Value
|  |  |  |  1: '000010'
|  |  Data Prefetch: None
|  |  Index Prefetch: None
|  Lock Intents
|  |  Table: Intent Share
|  |  Row  : Next Key Share
|  Return Data to Application
|  |  #Columns = 1
Return Data Completion

End of section
```

You can see that the cost is 25.045372.

Now let's create an index on empno and include the phoneno as a non-key index part:

```
>db2 create unique index emp_ix2 on employee (empno) include (phoneno)
```

Remember that if you want to include non-key data, the index you create must be unique.

Now, if we disconnect and reconnect from/to the database:

```
>db2 connect reset
>db2 connect to sample
```

and use **db2expln** again:

```
>db2expln -d sample -q "select phoneno from employee where empno =
'000010'"  -t
```

The output is:

```
SQL Statement:

  select phoneno
  from employee
  where empno = '000010'

Section Code Page = 1252

Estimated Cost = 0.037556
Estimated Cardinality = 1.000000

Access Table Name = DB2ADMIN.EMPLOYEE  ID = 2,5
|  #Columns = 1
|  Single Record
|  Index Scan:  Name = DB2ADMIN.EMP_IX2  ID = 2
|  |  Regular Index (Not Clustered)
|  |  Index Columns:
|  |  |  1: EMPNO (Ascending)
|  |  |  2: PHONENO (Include Column)
|  |  #Key Columns = 1
|  |  |  Start Key: Inclusive Value
|  |  |  |  1: '000010'
|  |  |  Stop Key: Inclusive Value
|  |  |  |  1: '000010'
|  |  Index-Only Access
|  |  Index Prefetch: None
|  |  Return Data to Application
|  |  |  #Columns = 1
|  Lock Intents
|  |  Table: Intent Share
|  |  Row  : Next Key Share
Return Data Completion

End of section
```

The cost is 0.037556 and you can see that the access is index only. Comparing the costs between using the indexes, you can see that the cost has been reduced from 25.045372 to 0.037556. If we try to add a second column to the query, such as firstnme, so that it looks like:

```
select firstnme,phoneno from employee where empno = '000010'
```

the cost goes back up to 25.045372 because we have to go to the data page to get the information for firstnme.

You can see the benefit of using non-key data in an index. However, if you are not sure which columns you want to retrieve, the overhead of creating many indexes and/or combinations of indexes may be prohibitive.

C Leonard
Freelance Consultant (UK)

SQL execution – revisited

In addition to the code in the article 'SQL execution' (see *DB2 Update*, issue 138, April 2004), here is @FTEP. This program can be used to format the online SQL output. It removes the column heading and separators. First you execute @SQL and choose on-line execution. The output will be displayed on your screen.

Additionally you can execute @FTEP. Only the data will be left.

@FTEP

```
* Rexx ************************************************************/
/* */
/* Format DSNTEP2-output */
/* */
/*******************************************************************/
'ISREDIT MACRO (DUMMY)'
/* trace r */
"Isredit (cursor) = user_state"
'ISREDIT RESET'
'ISREDIT RECOVER ON'
'ISREDIT BOUNDS 1 133'
'ISREDIT NULLS'
'ISREDIT NUM OFF'
'ISREDIT (CAPSTAT) = CAPS'
'ISREDIT CAPS OFF'
'ISREDIT EXCLUDE ALL'
'ISREDIT FIND " | " ALL'
'ISREDIT DELETE ALL EXCLUDED'
'ISREDIT EXCLUDE "_| " ALL'
'ISREDIT EXCLUDE " " FIRST'
```

```
'ISREDIT DELETE ALL NX'
'ISREDIT FIND "_| " FIRST'
'ISREDIT (REG,KOL) = CURSOR'
'ISREDIT FIND "_| "' kol 'ALL'
'ISREDIT FIND " " FIRST'
'ISREDIT DELETE ALL EXCLUDED'
'ISREDIT (LASTREG) = LINENUM .ZLAST'
do i=1 to LASTREG
'ISREDIT SHIFT (' i kol
'ISREDIT SHIFT )' i 1
end
'ISREDIT RESET'
'ISREDIT CHANGE "|" " " ALL'
'ISREDIT LINE_BEFORE 2 = " "'
'ISREDIT RESET'
'ISREDIT LINE_BEFORE 1 = MSGLINE "DSNTEP Uitvoer geformatteerd..."'
'ISREDIT CAPS' CAPSTAT
"Isredit user_state = (cursor)"
```

Loet J Polkamp
Database Administrator (The Netherlands)

Monitoring virtual storage in DB2 for OS/390 and z/OS

The DBM1 address space for each DB2 subsystem consists of storage above and below the 16MB line. Storage below the 16MB line is up to 16MB, while storage above the 16MB line can go up to 2GB or, more precisely, 2032MB. In this article, we will see what consumes this storage, a couple of methods for monitoring this storage, and a few steps we can take to reduce this storage. A REXX routine and some JCL are also provided to monitor and format the virtual storage information. With the new z/OS 64-bit machines with virtual storage support, and DB2 Version 8, it is expected that there will be fewer constraints on the use of virtual storage by DB2.

Typically, the storage below the line is used by the datasets that are open and each open dataset requires about 200 to 300 bytes. The storage above the line is consumed by DB2

code (about 25 to 30MB in DB2 V7), virtual buffer pools, EDM pool, RID pool, compression dictionaries, local dynamic SQL cache, RDS OP pool, storage required by various service stacks, system storage, agent local pool storage, control blocks for the virtual pool, and hiperpool buffers. Each virtual pool buffer page requires 128 bytes for the control block and each hiperpool buffer page requires 56 bytes for the control block. Theoretically the virtual buffer pools can go up 1.6GB. However, in reality the ceiling may be far less. This is because of all the other memory requirements listed above. The hiperpools are in extended storage and they max out at 8GB. They are a good proposition to use in certain kinds of application, but may cause excessive page swapping.

Guidelines regarding available virtual storage are as follows:

- Above 500MB free – OK.
- 200MB to 500MB free – warning, close monitoring required.
- Below 200MB free – action required.

Whenever DB2 faces a shortage on storage, it will try to recover from the related abend using the recovery routines. However, if the task being performed is very critical and no errors can be tolerated, the DB2 subsystem will crash.

There are three ways in which the current storage utilization may be monitored:

1 Using the DB2 instrumentation facility interface (IFCID 225/IFCID 217).

2 Using the RMF monitor.

3 Using the DB2 dump formatter, DSNWDMP.

The first two methods will be discussed in this article. The third method may be used to analyse the actual virtual storage at the time DB2 terminated abnormally. For information on using the DSNWDMP program, please refer to the *DB2 Diagnosis Guide and Reference*.

USING THE DB2 INSTRUMENTATION FACILITY INTERFACE

New instrumentation was introduced in DB2 V7 to monitor virtual storage. IFCID 217 will monitor storage usage at a detailed level, whereas IFCID 225 is at the summary level. For most purposes IFCID 225 is adequate and that is discussed here in detail.

The IFCID 225 record is generated through the statistics trace, and it is generated as an SMF type 102 record. To minimize the impact of running this trace, start it as follows:

```
-START TRACE(STAT) DEST(SMF) CLASS(6) IFCID(Ø225)
```

The trace record is generated at the frequency at which the statistics trace record is generated and is determined by a system parameter. Normally, it is 10 or 15 minutes. After, say, about 30 minutes, terminate the trace using this command:

```
-STO TRACE(STAT) CLASS(6) DEST(SMF) TNO(x)
```

where *x* is the trace number that was indicated when the trace was started.

Normally the DB2 trace data that is sent to SMF is aggregated into a separate file and is kept in a GDG dataset. You will have to find out the relevant dataset that contains your trace data based on your installation standards and use that in the JCL given below:

```
//Your Job card
//*
//* USE RECFM=VB TO READ THE DATASET WITH ISPF OR REXX
//*
//SORTS1   EXEC PGM=ICEMAN
//SYSOUT   DD   SYSOUT=*
//*
//SORTIN   DD DISP=SHR,DSN=The SMF Trace dataset
//*
//SORTOUT  DD   DSN=The output dataset
//         DISP=(NEW,KEEP),
//         UNIT=HSM,
//         DCB=(RECFM=VB,LRECL=32756,BLKSIZE=3276Ø,DSORG=PS),
//         SPACE=(CYL,(5Ø,1Ø),RLSE)
//SYSIN    DD    *
  SORT FIELDS=(7,4,CH,A)
```

```
 INCLUDE COND=(6,1,CH,EQ,X'66')
/*
//*
//SORTWKØ1 DD   UNIT=SYSWK,SPACE=(CYL,(2ØØ,5Ø),RLSE)
//SORTWKØ2 DD   UNIT=SYSWK,SPACE=(CYL,(2ØØ,5Ø),RLSE)
//SORTWKØ3 DD   UNIT=SYSWK,SPACE=(CYL,(2ØØ,5Ø),RLSE)
//SORTWKØ4 DD   UNIT=SYSWK,SPACE=(CYL,(2ØØ,5Ø),RLSE)
//*
//
```

The JCL shown uses the SORT program ICEMAN. DFSORT may also be used to achieve the same results. The SORTIN dataset will be the GDG or any other dataset that has the relevant trace data that we captured. The SORTOUT dataset will be the output dataset with only SMF type 102 records. After successful completion of this job, run the REXX, STOREP, shown below, with this dataset as input:

```
/* Rexx Exec */
pref =strip(sysvar(syspref))
NUMERIC DIGITS 2Ø
PARSE UPPER ARG P_dsname
if strip(P_dsname)='' then
do
   Call GETDBLST
end
else
do
   I_lstdsn = strip(P_dsname)
   I_lstdsn = strip(P_dsname,B,"'")
end
Call ALLOCDSN
do i=1 to dbl.Ø
   trace_line = dbl.i
   call format_record
end
exit Ø
format_record:
/* QWSP  */
   product_offset = c2d(substr(trace_line,25,4))
   product_length = c2d(substr(trace_line,29,2))
   product_repeat = c2d(substr(trace_line,31,2))
   ifcid = c2d(substr(trace_line,product_offset+1,2))
   if ifcid = 225 then
   do
      ssid = (substr(trace_line,product_offset+9,4))
      stck_offset = product_offset + 13
      stck = c2x(substr(trace_line,stck_offset,8))
```

```
      Call STCKCONV
      say "SSID     DATE          TIME   "
      say "-------------------------------"
      say left(ssid,5) left(dat,11) tim
/*    say 'IFCID DATE TIME  SSID STMT' */
      say
      call format_225
   end
return
format_225:
   data1_offset = c2d(substr(trace_line,33,4))-3
   data1_length = c2d(substr(trace_line,37,2))-4
   data1_repeat = c2d(substr(trace_line,39,2))
qw0225_lgth = c2d(substr(trace_line,data1_offset+1,2))-2
head_detail = substr(trace_line,data1_offset+3,qw0225_lgth)
say  "IFCID 225 storage pool stats follow"
say  "***********************************"
QW0225AL= trunc((c2d(substr(trace_line,data1_offset,4))/ 1048576),2)
QW0225AS= trunc((c2d(substr(trace_line,data1_offset+4,4)) / 1048576),2)
QW0225AV= trunc((c2d(substr(trace_line,data1_offset+8,4)) / 1048576),2)
QW0225CD= trunc((c2d(substr(trace_line,data1_offset+12,4)) / 1048576),2)
QW0225CR= trunc((c2d(substr(trace_line,data1_offset+16,4)) / 1048576),2)
QW0225FX= trunc((c2d(substr(trace_line,data1_offset+20,4)) / 1048576),2)
QW0225GM= trunc((c2d(substr(trace_line,data1_offset+24,4)) / 1048576),2)
QW0225GS= trunc((c2d(substr(trace_line,data1_offset+28,4)) / 1048576),2)
QW0225MV= trunc((c2d(substr(trace_line,data1_offset+32,4)) / 1048576),2)
QW0225PM= trunc((c2d(substr(trace_line,data1_offset+36,4)) / 1048576),2)
QW0225RO= trunc((c2d(substr(trace_line,data1_offset+40,4)) / 1048576),2)
QW0225RP= trunc((c2d(substr(trace_line,data1_offset+44,4)) / 1048576),2)
QW0225SB= trunc((c2d(substr(trace_line,data1_offset+48,4)) / 1048576),2)
QW0225SC= trunc((c2d(substr(trace_line,data1_offset+52,4)) / 1048576),2)
QW0225SO= trunc((c2d(substr(trace_line,data1_offset+56,4)) / 1048576),2)
QW0225TT= trunc((c2d(substr(trace_line,data1_offset+60,4)) / 1048576),2)
QW0225VR= trunc((c2d(substr(trace_line,data1_offset+64,4)) / 1048576),2)
QW0225AT= c2d(substr(trace_line,data1_offset+68,4))
QW0225CE= c2d(substr(trace_line,data1_offset+72,4))
QW0225DW= c2d(substr(trace_line,data1_offset+76,4))
QW0225GW= c2d(substr(trace_line,data1_offset+80,4))
QW0225PF= c2d(substr(trace_line,data1_offset+84,4))
QW0225PL= c2d(substr(trace_line,data1_offset+88,4))
say " TOTAL AGENT LOCAL POOL STORAGE      :" right(QW0225AL,8) "MB"
say " TOTAL AGENT SYSTEM STORAGE          :" right(QW0225AS,8) "MB"
say " AMOUNT OF AVAIL STORAGE             :" right(QW0225AV,8) "MB"
say " TOTAL COMPRESS DICTIONARY STORAGE   :" right(QW0225CD,8) "MB"
say " STG RSRVD ONLY FOR MUST COMPLETE    :" right(QW0225CR,8) "MB"
say "TOTAL FIXED STORAGE                  :" right(QW0225FX,8) "MB"
say "TOTAL GETMAINED STORAGE              :" right(QW0225GM,8) "MB"
say "TOTAL GETMAINED STACK STORAGE        :" right(QW0225GS,8) "MB"
say " AMOUNT OF STORAGE FOR MVS USAGE     :" right(QW0225MV,8) "MB"
say " TOTAL PIPE MANAGER SUBPOOL STORAGE  :" right(QW0225PM,8) "MB"
```

```
say " TOTAL RDS OP POOL STORAGE           :" right(QWØ225RO,8) "MB"
say " TOTAL RID POOL STORAGE              :" right(QWØ225RP,8) "MB"
say " TOTAL STATEMENT CACHE BLOCK STORAGE :" right(QWØ225SB,8) "MB"
say " TOTAL STORAGE FOR THREAD COPIES .... "
say "  OF CACHED SQL STATEMENTS           :" right(QWØ225SC,8) "MB"
say "STG CUSHION WARNING TO CONTRACT      :" right(QWØ225SO,8) "MB"
say " TOTAL BM/DM INTERNAL TRACE TBL STOR :" right(QWØ225TT,8) "MB"
say "TOTAL VARIABLE STORAGE               :" right(QWØ225VR,8) "MB"
say " # OF ACTIVE ALLIED THREADS          :" right(QWØ225AT,8)
say " # OF CASTOUT ENGINES                :" right(QWØ225CE,8)
say " # OF DEFERRED WRITE ENGINES         :" right(QWØ225DW,8)
say " # OF GBP WRITE ENGINES              :" right(QWØ225GW,8)
say " # OF PREFETCH ENGINES               :" right(QWØ225PF,8)
say " # OF P-LOCK/NOTIFY EXIT ENGINES     :" right(QWØ225PL,8)
say "-------------------------------------------------"
say
return
STCKCONV:
trace off
tsec = (substr(stck,1,13))     /* Ignore last 1.5 bytes     */
NUMERIC DIGITS 2Ø
secs = X2D(tsec)
micsec = ((secs / 1ØØØØØØ) - (secs % 1ØØØØØØ))* 1ØØØØØØ
micsec = trunc(micsec,Ø)
secs = (secs - micsec)%1ØØØØØØ
/* offset by 5 hours for GMT */
secs = secs - (5*36ØØ)
year = secs / (36ØØ*24*365.25)
addon = trunc(year,Ø) / 4 - trunc(year,Ø) % 4 /* take care of leap yr*/
jd   = (year - trunc(year,Ø)) * 365.25 + addon
if addon = Ø then jd = jd + 1    /* leap year */
hours = (jd  - trunc(jd,Ø) ) * 24
min   = (hours - trunc(hours,Ø)) * 6Ø
sec   = (min - trunc(min,Ø))*6Ø
prec = sec - trunc(sec,Ø)
if prec > Ø.98 then sec = sec + 1
tim = right('ØØ'||trunc(hours,Ø),2) || '.' || right('ØØ'||,
 trunc(min,Ø),2) || '.' || right('ØØ'||trunc(sec,Ø),2) || '.' ||micsec
NUMERIC DIGITS 9               /* restore default significant digits */
year = trunc(year,Ø) + 19ØØ      /*  add 19ØØ-Ø1-Ø1  */
jd   = trunc((jd+1),Ø)                /* to current date     */
jdcal          = '031059090120151181212243273304334365'
jdcal_leap =     '031060091121152182213244274305335366'
mon = 1
prev_mon_days = Ø
do forever
  start_pos = (mon-1)*3 + 1
  if addon = Ø then days = substr(jdcal_leap,start_pos,3)
  else days = substr(jdcal,start_pos,3)
  if jd < days | jd = days then
```

```
    do
      days = jd - prev_mon_days
      leave
    end
    mon = mon + 1
    prev_mon_days = days
end
dat = year|| '-' || Right('ØØ'||mon,2) || '-' || Right('ØØ'||days,2)
NUMERIC DIGITS 2Ø
return

GETDBLST:
Say  'Give the input dataset ...'
Say  '(It must be a PS )'
pull I_lstdsn
/* I_lstdsn = HRXS.$$PAJKJ.OUTPUT.DMPTDBT3.D99Ø2Ø8 */
I_lstdsn = strip(I_lstdsn)
I_lstdsn = strip(I_lstdsn,Both,"'")
x = SYSDSN("'"I_lstdsn"'")
if x ¬= OK then
do
   say; say '*** ERROR ' x ; say
   SIGNAL  GETDBLST
end
return

ALLOCDSN:
"ALLOCATE DD(lstdd) DSN('"I_lstdsn"') REUSE SHR"
if  rc>Ø then
do
   say  'Failed during allocation of 'I_lstdsn
   exit(8)
end
"execio * DISKR lstdd (FINIS STEM dbl."

address tso "free f(lstdd)"
return
```

The output will be displayed on the screen as shown in the sample below:

```
SSID     DATE          TIME
-------------------------------
PROD  2ØØ3-12-31  Ø9.Ø4.26.972576

IFCID 225 storage pool stats follow
***********************************
 TOTAL AGENT LOCAL POOL STORAGE      :    61.63 MB
 TOTAL AGENT SYSTEM STORAGE          :    29.78 MB
 AMOUNT OF AVAIL STORAGE             :    76.18 MB
```

```
 TOTAL COMPRESS DICTIONARY STORAGE   :    59.09 MB
 STG RSRVD ONLY FOR MUST COMPLETE    :    25.41 MB
TOTAL FIXED STORAGE                  :     3.79 MB
TOTAL GETMAINED STORAGE              :  1105.10 MB
TOTAL GETMAINED STACK STORAGE        :    32.10 MB
 AMOUNT OF STORAGE FOR MVS USAGE     :     1.08 MB
 TOTAL PIPE MANAGER SUBPOOL STORAGE  :     0.00 MB
 TOTAL RDS OP POOL STORAGE           :     0.96 MB
 TOTAL RID POOL STORAGE              :     0.09 MB
 TOTAL STATEMENT CACHE BLOCK STORAGE :    39.22 MB
 TOTAL STORAGE FOR THREAD COPIES ....
  OF CACHED SQL STATEMENTS           :     0.53 MB
STG CUSHION WARNING TO CONTRACT      :    79.19 MB
 TOTAL BM/DM INTERNAL TRACE TBL STOR :    10.35 MB
TOTAL VARIABLE STORAGE               :   121.48 MB
 # OF ACTIVE ALLIED THREADS          :      101
 # OF CASTOUT ENGINES                :        0
  # OF DEFERRED WRITE ENGINES        :      300
  # OF GBP WRITE ENGINES             :        0
  # OF PREFETCH ENGINES              :      120
  # OF P-LOCK/NOTIFY EXIT ENGINES    :        0
```

At our installation, the STORCLOCK is set to GMT, and hence the REXX has made an adjustment to convert it to local time. This may be modified appropriately to suit the installation settings and also the local time difference with reference to GMT.

In the sample output the *AMOUNT OF AVAIL STORAGE* field shows the amount of free virtual storage available. Depending on the volume of DB2 threads, we can predict when DB2 is likely to crash. When this value approaches 100MB, we will have either to recycle DB2 or make drastic measures to reduce storage consumption, like reducing virtual bufferpool, EDM pool, etc. When DB2 is about to go down, we see DSNL027I messages with abend code 04E and reason code 00E20003 for distributed (DDF) threads. For CICS and batch threads we see the DSN3201I message before DB2 terminates. Eventually DB2 may terminate with the*DSNV086E message with reason code 00E50727.

USING THE RMF MONITOR

Using RMF, we can monitor the DBM1 address space of a

particular DB2 subsystem. Using the RMF batch post processor, we can execute a report against the SMF type 78 subtype 2 records collected in the RMF monitor session. The command to start RMF monitoring is given below (it requires authorization to execute console commands):

```
F RMF,F ZZ,VSTOR(D,PRODDBM1)
```

The above command initiates monitoring of the PRODDBM1 address space belonging to a DB2 subsystem named PROD. This command will generate a message in the SYSLOG similar to the one given below:

```
syst     Ø3365 1Ø:ØØ:Ø9.41 STC26148 ØØØØØØ9Ø  ERB1Ø4I ZZ : MODIFIED
```

Based on RMF and SMF parameters in the SYS1.PARMLIB dataset the collection frequency will vary. After two or three intervals have elapsed, terminate the RMF monitoring with the following command:

```
F RMF,F ZZ,NOVSTOR
```

This is very critical because if the VSTOR data gathering function is left on for an extended period of time, it might affect system performance. After the data gathering is completed, we can generate a report using the sample JCL below:

```
// job card
//*
//*
//SORTRMF      EXEC PGM=SORT,PARM='SIZE(4ØØØØØ)'
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SORTWKØ1 DD  UNIT=SYSDA,SPACE=(CYL,(2ØØ,5Ø))
//SORTWKØ2 DD  UNIT=SYSDA,SPACE=(CYL,(2ØØ,5Ø))
//SORTWKØ3 DD  UNIT=SYSDA,SPACE=(CYL,(2ØØ,5Ø))
//SORTWKØ4 DD  UNIT=SYSDA,SPACE=(CYL,(2ØØ,5Ø))
//SORTIN   DD   DISP=OLD,DSN=The RMF data collection dataset
//SORTOUT  DD  DSN= the output dataset,
//         UNIT=HSM,
//         SPACE=(CYL,(3ØØ,1ØØ),RLSE),
//         DCB=(RECFM=VBS,LRECL=32756,BLKSIZE=4Ø96),
//         DISP=(NEW,CATLG,DELETE)
//SYSIN DD *
 SORT FIELDS=(15,2,CH,A,11,4,CH,A,7,4,CH,A),FILSZ=E4ØØØØØ
 RECORD TYPE=V
```

```
 OMIT COND=(6,1,BI,LT,X'46',OR,6,1,BI,GT,X'4F')
/*
//*
//RMFREPT  EXEC PGM=ERBRMFPP,REGION=8M,COND=(Ø,NE)
//STEPLIB  DD  DSN=CEE.SCEERUN,DISP=SHR
//MFPMSGDS DD SYSOUT=*
//MFPINPUT DD DSN=The output dataset from previous step,DISP=OLD
//SYSIN    DD *
  REPORTS(VSTOR(D,PRODDBM1))
//
```

Sample output is shown below:

```
1        V I R T U A L   S T O R A G E   A C T I V I T Y

                                    PAGE    3
    OS/39Ø                   SYSTEM ID SYSB             DATE 12/31/2ØØ3
      INTERVAL 14.44.297
    REL. Ø2.1Ø.ØØ            RPT VERSION Ø2.1Ø.ØØ       TIME 1Ø.ØØ.15
      CYCLE 1.ØØØ SECONDS
 LSQA/SWA/229/23Ø
  FREE PAGES (BYTES)     ØK 1Ø.ØØ.16     ØK              ØK  ****K
1Ø.13.45  2Ø11M 1Ø.Ø9.26  1645M
  LARGEST FREE BLOCK     ØK 1Ø.ØØ.16     ØK              ØK     ØK
1Ø.ØØ.16     ØK              ØK
  PAGES ALLOCATED
   (IN BYTES)          792K 1Ø.ØØ.16   792K 1Ø.ØØ.16   792K   765M
1Ø.Ø4.26   765M 1Ø.ØØ.16   765M
ØUSER REGION
  FREE PAGES (BYTES)  9224K 1Ø.ØØ.16  9224K 1Ø.ØØ.16  9224K  ****K
1Ø.Ø9.26  1851M 1Ø.13.45  1892M
  LARGEST FREE BLOCK
    IN GETMAIN LIMIT  9Ø16K 1Ø.ØØ.16  9Ø16K 1Ø.ØØ.16  9Ø16K  1635M
1Ø.ØØ.16  1635M 1Ø.ØØ.16  1635M
  PAGES ALLOCATED
    (IN BYTES)         2Ø4K 1Ø.ØØ.16   2Ø4K 1Ø.ØØ.16   2Ø4K  35.4M
1Ø.ØØ.16  35.4M 1Ø.ØØ.16  35.4M
```

To obtain the amount of free virtual storage, get the following figures. Look under the heading *LSQA/SWA/229/230* and find the *Pages allocated (in bytes)* subheading. Choose the *AVG* value against this (765M in the example) and call this *LSQA Pages*. A few lines further down, under the heading *User Region*, find the *Pages Allocated (in bytes)* subheading and choose the *AVG* value against this (35.4M in the example). Call this *User region pages*. At the top of the report find the *Region assigned above 16M* amount (1669M in the example)

and call this *Region assigned*. The free virtual storage available is calculated as:

```
Region assigned – (LSQA Pages + User region pages)
```

In the sample report shown, this value will be:

```
1669 – (765 + 34.5)  = 868.6MB
```

CONCLUSION

Utilizing the above methods, we can monitor the virtual storage of the DB2 DBM1 address space and take corrective action as required. There are numerous fixes that are released to address storage issues in DB2. Storage INFO APARS for DB2 II04309 and II10817 contain information about those fixes.

Some of the ways to alleviate virtual storage issues are as follows.

Recycling the DB2 subsystem on a regular basis where possible will help to a large extent. Where this is not feasible, the most effective action will be to reduce the virtual bufferpool and increase the hiperpool allocations. Setting the CONTSTOR zparm to YES helps to regain storage in conjunction with two other parameters, SPRMSTH (threshold before contraction is performed) and SPRMCTH (number of commits before a contraction is performed). Decreasing the local dynamic SQL cache by reducing the MAXKEEPD zparm may help, as might reducing the DSMAX zparm to reduce the number of datasets that are open. Implementing dataspaces for EDM pools and bufferpools could be another alternative to reduce virtual storage constraints. Use of dataspaces for EDM pools is preferable to use of dataspaces for virtual pools.

Version 8 of DB2 with 64-bit support on the z/OS machines promises to alleviate the virtual storage issues. In any case, by routinely monitoring virtual storage we can take pre-emptive action to keep the DB2 subsystem healthy and avoid unexpected crashes.

REFERENCES

1 *Virtual Storage Management Before and After zSeries*, John J Campbell, IBM Data Management Technical Conference, September 2003.

2 *Monitoring Virtual Storage Consumption*, John E Johnston, *z/Journal* December 2003/January 2004.

Jaiwant Jonathan
Senior Consultant
QSS Inc (USA)

Spiffing up SPUFI (Version 2) – part 2

This month we conclude the improved version of invoking SPUFI from EDIT mode on SQL.

```
         Otherwise                    /* longer line, must be split */
             reps = line_leng % 10
             If cs = 1,              /* comment starting at column 1 */
              & Left(line,10*reps) = Copies('--------+',reps) Then
                input.j = Left(line,80)     /* SPUFI output comment */
             Else Do
                Call REFORMAT_LINE
                If result > 0 Then
                   Return 16         /* message about the error */
                End
           End
         End
      /*-- end of input, check that all char are strings complete --*/
      If quote <> ' ' Then Do
         Call INVALID_QUOTES_MSG
         Return 16
         End
      End
   /*-- write input summary into title lines --*/
   If sql_title = 'YES' Then Do
      If reformat = 'NO' Then
         input_summary = input_summary ,
            'and first 80 bytes used with no reformatting'
      If reformat = 'YES' Then Do
         If lines_split = 0
           Then ref_sum = 'all SQL was already in first 72 bytes'
```

```
            Else ref_sum = lines_split 'reformatted into first 72 bytes'
         input_summary = input_summary 'and' ref_sum
         End
      If esql_mode = 'BATCH'
         Then input.4 = input_summary
         Else input.3 = input_summary
      End
   input.Ø = j
   Return Ø
/*====================================================================*/
/* find the start of any comment on this line                         */
/*--------------------------------------------------------------------*/
FIND_COMMENT:
   cs = 1
   Do k = 1 to 9999 While cs > Ø
      cs = Pos('--',line,cs)        /* look for start of a comment */
      If cs > Ø Then Do
         ncs = IN_STRING(cs,'END')
         If ncs = Ø Then Leave k        /* not in a string */
         If ncs > Ø Then cs = ncs + 1   /* look again after string */
         If ncs = -1 Then Return 16  /* invalid: incomplete string */
         End
      End
   Return Ø
/*====================================================================*/
/* remove a line number (if it exists)                                */
/*--------------------------------------------------------------------*/
REMOVE_LINE_NUMBER:
   /* num = EDIT auto-numbering: 'ON' or 'OFF' */
   /*-- look for an old line number --*/
   If line_leng = lrecl & num = 'OFF' & fm1 = 'F' Then Do
      Do chk_posn = lrecl To lrecl-7 By -1
         dt = Datatype(Substr(line,chk_posn,1))
         If dt <> 'NUM' Then Leave
         End
      End
   Else dt = ''
   /*-- remove line number --*/
   If num = 'ON' | dt = 'NUM' Then Do      /* there is a line number */
      If fm1 = 'F' Then Do
         data_leng = lrecl - 8
         line = Left(line,data_leng)        /* remove numbers at end */
         End
      If fm1 = 'V' Then
         line = Substr(line,9)            /* remove numbers at start */
      End
   line = Strip(line,'T')              /* remove any trailing blanks */
   line_leng = Length(line)
   Return
/*====================================================================*/
```

```
/* reformat a long input line into multiple pieces which will fit    */
/* into RECFM=F,LRECL=8Ø input for SPUFI or DSNTEP2                  */
/*-------------------------------------------------------------------*/
REFORMAT_LINE:
   no_comment = 1
   l_end = Ø                     /* end column (in line) of the piece */
   lines_split = lines_split + 1
   /*-- process line, splitting it into up to 9999 pieces! --*/
   Do pieces = 1 To 9999
      line_leng = length(line)
      If no_comment Then Do
         /*-- check for a comment in the next piece --*/
        If Ø < cs & cs < l_end + 72 Then Do /* cs = start of comment */
            no_comment = Ø                       /* a comment starts */
           /* comments can go up to column 72 (batch) or 8Ø (SPUFI)  */
            If line_leng < end_comment + 1
               Then end_col = end_comment  /* end_comment = 72 or 8Ø */
               Else end_col = Lastpos(' ',Left(line,end_comment))
            End
         /*-- splitting up SQL (with no comment) --*/
         Else Do
            sql_text = line
            end_col = 72   /* SQL up to 72 bytes per line for SPUFI */
            /* split at last ')',' ' or ',' that's not in a string  */
            If line_leng > 72 Then Do
               c = end_col
               Do split = 1 to 5Ø While c > Ø
                  c1 = Lastpos(')',line,c)
                  c2 = Lastpos(' ',line,c)
                  c3 = Lastpos(',',line,c)
                  c = Max(c1,c2,c3)
                  If c > 1 Then Do
                     nc = IN_STRING(c,'START')
                     If nc = Ø & quote = '' Then  /* not in string */
                        Leave split           /* we can split here */
                     If nc > 2 Then Do
                        c = nc - 1     /* look again before string */
                        Iterate split
                        End
                     If nc = -1 Then Return 16
                     /* nc = 1 or 2, or (nc = Ø & quote <> '') */
                     c = Ø
                     End
                  End    /* end of split */
               If c > 1 Then end_col = c
               End     /* end of line_leng > 72 */
            /*-- check for unfinished string --*/
            sql_text = Left(line,end_col)
            lch = IN_STRING(73,'START')  /* unfinished string? */
            cont_quote = quote
```

```
            End
         l_end = l_end + end_col
         input.j = Left(line,end_col)
         If no_comment | esql_mode = 'BATCH' Then
            input.j = Overlay('<-REFORM',input.j,73)
         End
      /*-- continuing a comment from a previous piece --*/
      Else Do
         end_col = Lastpos(' ',Left(line,77))
         input.j = '--' Left(line,end_col)
         If esql_mode = 'BATCH' Then
            input.j = Overlay('<-REFORM',input.j,73)
         End
      /*-- prepare line for the next iteration of pieces --*/
      If line_leng > end_col Then Do
         line = Substr(line,end_col+1)
         sql_text = line
         End
      Else Leave pieces  /* no more SQL in line - we're finished */
      If line = '' Then Leave pieces
      j = j + 1
      End
   Return
/*====================================================================*/
/* is the specified character in a quoted string?                     */
/*  Returns: Ø   not in a string                                      */
/*           -1  in string, but no closing quote on this line         */
/*           nn  position of start/end quote of string that it's in   */
/*--------------------------------------------------------------------*/
IN_STRING:
trace O
   Arg p,dir  /* column, 'START' or 'END' */
   quote = cont_quote
   If cont_quote = ''
      Then sq = 1
      Else sq = Ø  /* continuing string from prev (SPUFI input) line */
   Do forever
      /*-- look for starting quote of a string --*/
      If sq > Ø Then Do
         quote = ''
         q1 = Pos("'",sql_text,sq)      /* look for single quotes */
         q2 = Pos('"',sql_text,sq)      /* look for double quotes */
         Select
           When q1 + q2 = Ø Then Return Ø         /* no quotes found */
           When q1 > Ø & (q1 < q2 | q2 = Ø) Then Do
              sq = q1
              If sq > p Then Return Ø        /* 1st quote is after p */
              quote = "'"
              End
           When q2 > Ø & (q2 < q1 | q1 = Ø) Then Do
```

```
            sq = q2
            If sq > p Then Return Ø        /* 1st quote is after p */
            quote = '"'
            End
          End
        End
      /*-- look for ending quote of a string --*/
      eq = Pos(quote,sql_text,sq+1)   /*look for next (ending) quote */
      Select
        When eq = Ø Then Do                 /* no ending quote found */
          If line_leng > 72 & Length(sql_text) > 72 Then
            Call INVALID_QUOTES_MSG('on or before line' i)
          Return -1
          End
        When eq > p Then Do      /* position p is inside a string .. */
          If dir = 'END' Then Return eq   /* quote at end of string */
          If dir = 'START' Then Return sq  /* pos'n of string start */
          End
        Otherwise             /* string ended (eq) before position p */
          quote = ''
          sq = eq + 1
        End
      End     /* end of Do forever */
   Return Ø
/*====================================================================*/
/* write ISPF message to warn that unmatched quotes found             */
/*--------------------------------------------------------------------*/
INVALID_QUOTES_MSG:
   Parse Arg msg_text
   If msg_text <> '' Then Do
      Address ISREDIT "UP MAX"
      Address ISREDIT "DOWN" i-1
   /* Say 'unmatched quote =' quote ',' sql_text */
      End
   ZERRLM = 'SQL INVALID - unmatched quotes found' msg_text
   ZERRHM = '*'
   ZERRALRM = 'YES .WINDOW=LNORESP'
   Address ISPEXEC "SETMSG MSG(ISRZØØ2)"
   Return
/*====================================================================*/
/* write specified SQL into input dataset for SPUFI                   */
/*--------------------------------------------------------------------*/
CREATE_SPUFIDSN:
   If spufidsn = '' Then Do
      /*-- set name of SPUFI input dataset which will be created --*/
      spufidsn = Userid()||'.SPUFIN'
      tsopref = Sysvar(SYSPREF)
      /* default TSO PREFIX here is same as the user's RACF ACCOUNT */
      If tsopref = '' Then Do    /* if user has TSO PROFILE NOPREFIX */
         w = Outtrap('racfmsg.')
```

```
         Address TSO "LU "||Userid()||" NORACF TSO" /* RACF TSO info */
         Parse Var racfmsg.5 . tsopref .       /* 'ACCTNUM= tsopref' */
         x = Outtrap('OFF')
         End
      If tsopref <> Userid() & tsopref <> '' Then
         spufidsn = tsopref||'.'||spufidsn
      End
   /*-- allocate the SPUFI input dataset --*/
   oldstat = MSG('OFF')              /* ensure TSO messages are OFF */
   Address TSO "ALLOC F(SPUFIN) DSN('"spufidsn"') SHR REUSE"
   If rc > 0 Then Do
      Address TSO "ALLOC F(SPUFIN) DSN('"spufidsn"') NEW CATALOG" ,
         "SPACE(1,1) CYL RECFM(F B) LRECL(80) BLKSIZE(0)"
      alloc_rc = rc
      End
   a = MSG(oldstat)                  /* reinstate TSO messages      */
   /*-- copy SQL into the SPUFI input dataset --*/
   Address TSO "EXECIO * DISKW SPUFIN (STEM input. FINIS"
   If rc = 0 Then Do
      dset = "'"spufidsn"'"   /* dset non-null only when copy was OK */
      /*-- edit SPUFI input dataset (before it's passed to SPUFI) --*/
      If spufi_ed = 'YES' Then Do
         If spufi_pnl = 'YES' Then
            ZEDLMSG = '       ******  After the EDIT this filename',
                      'will be on the SPUFI panel  ******'
         Else
            ZEDLMSG = '       ******  This SQL will be processed by',
                      'SPUFI when this EDIT ends  ******'
         Address ISPEXEC "SETMSG MSG(ISRZ001)"
         Address ISPEXEC "EDIT DATASET('"spufidsn"')"
         End
      /*-- set name of SPUFI output dataset --*/
      If spufiout = '' Then Do    /* if name not specified in parms */
         If spufi_out = 'YES' Then spufiout =,
            "'"Left(spufidsn,Length(spufidsn)-1)||'OUT.'||ssid"'"
         If spufi_out = 'COND' Then spufiout =,
            "'"Left(spufidsn,Length(spufidsn)-1)||'OUT.'||ssid"',COND"
         If spufi_out = 'NO' Then spufiout = ''
         End
      End
   Return
/*====================================================================*/
/* run SPUFI online via the EXSPUFI EXEC                              */
/*--------------------------------------------------------------------*/
RUN_EXSPUFI:
   /*-- create parameter list for EXSPUFI --*/
   ex_parms = "ALLOC("libry_alloc")",     /* alloc DB2 libs   */
              "SSID("ssid")",             /* DB2 sub-system   */
              "INDSN("dset")",            /* input dataset    */
              "OUTDSN("spufiout")"        /* output dataset   */
```

```
   If spufi_pnl <> '' Then ex_parms = ex_parms ,
              "PANEL("spufi_pnl")"        /* show SPUFI panel */
   If spufi_loc <> '' Then ex_parms = ex_parms ,
              "LOC("spufi_loc")"          /* connect location */
   If spufi_len <> '' Then ex_parms = ex_parms ,
              "LRECL("spufi_len")"        /* output LRECL     */
   If spufi_blk <> '' Then ex_parms = ex_parms ,
              "BLK("spufi_blk")"          /* output BLKSIZE   */
   If spufi_fmt <> '' Then ex_parms = ex_parms ,
              "RECFM("spufi_fmt")"        /* output RECFM     */
   If spufi_uni <> '' Then ex_parms = ex_parms ,
              "UNIT("spufi_uni")"         /* output UNIT      */
   ex_parms = ex_parms ,
              "BR("spufi_br")"            /* browse output    */
   If spufi_iso <> '' Then ex_parms = ex_parms ,
              "ISO("spufi_iso")"          /* SPUFI isolation  */
   If spufi_mno <> '' Then ex_parms = ex_parms ,
              "MAXNO("spufi_mno")"        /* max num chars    */
   If spufi_mch <> '' Then ex_parms = ex_parms ,
              "MAXCH("spufi_mch")"        /* max char field   */
   If spufi_max <> '' Then ex_parms = ex_parms ,
              "MAXSEL("spufi_max")"       /* max SELECT lines */
   If spufi_col <> '' Then ex_parms = ex_parms ,
              "COL("spufi_col")"          /* column headings  */
   If spufi_com <> '' Then ex_parms = ex_parms ,
              "COM("spufi_com")"          /* auto-commit      */
   /*-- invoke EXSPUFI in the same logical screen (over the top)  --*/
   If spufi_screen = 'ONTOP' Then
      Address ISPEXEC "SELECT CMD(%EXSPUFI" ex_parms ")",
                      "NEWAPPL("ssid") PASSLIB"
   /*-- use program ISPSTRT to open a new logical (split) screen ----*/
   /*-- this only works if EXSPUFI in SYSEXEC or SYSPROC library ----*/
   /*-- (note that maximum allowed PARM length is 249 bytes)     ----*/
   If spufi_screen = 'NEW' Then
      Address ISPEXEC "SELECT PGM(ISPSTRT)",
                      "PARM(CMD(%EXSPUFI" ex_parms ")",
                      "NEWAPPL("ssid") SCRNAME(SPUF"ssid"))"
   /*-- delete the SPUFI input file --*/
   If alloc_rc = Ø Then Do
      oldstat = MSG('OFF')
      Address TSO "FREE FILE(SPUFIN)"
      Address TSO "DELETE '"spufidsn"'"
      a = MSG(oldstat)
      End
   Return
/*====================================================================*/
/* create JCL for a batch job, and submit it                          */
/*--------------------------------------------------------------------*/
SUBMIT_BATCH_JOB:
   /*-- set character to be used in jobname --*/
```

```
Address ISPEXEC "VGET ESQLCHAR PROFILE"
If ESQLCHAR = '' Then ESQLCHAR = ' '
ESQLCHAR = Translate(ESQLCHAR,  /* change it to the next character*/
            ,'123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ1', /* out */
            ,' 123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ') /* in  */
Address ISPEXEC "VPUT ESQLCHAR PROFILE"   /* save it for next time*/
/*-- ISPF file tailoring to create batch job JCL --*/
Address ISPEXEC "FTOPEN TEMP"
Address ISPEXEC "FTINCL ESQLJOB"
incl_rc = rc
If ZERRMSG <> 'ZERRMSG' Then
   Address ISPEXEC "SETMSG MSG("ZERRMSG")"   /* show standard msg */
Address ISPEXEC "FTCLOSE"
If incl_rc > Ø Then Return
/*-- copy SQL to the end of the JCL dataset --*/
Address ISPEXEC "VGET (ZTEMPN ZTEMPF)"
Address TSO "ALLOC FI("ZTEMPN") DSN('"ZTEMPF"') MOD REUSE"
Address TSO "EXECIO * DISKW" ZTEMPN "(STEM input. FINIS"
Address TSO "FREE FI("ZTEMPN")"
/*-- submit the job --*/
If edit_jcl = 'YES' Then
   Address ISPEXEC "EDIT DATASET('"ZTEMPF"')"
If submit_jcl = 'YES' Then
   Address TSO "SUBMIT '"ZTEMPF"'"
Return
```

EXSPUFI EXEC

This doesn't need changing.

```
/*============================> REXX <================================*/
/* EXSPUFI: Execute SPUFI                                 Version 2   */
/*--------------------------------------------------------------------*/
/*      a) Command ('ESQL' or 'ESQL ssid') is entered on the command */
/*         line of an EDIT/VIEW of some SQL.  ESQL edit macro runs   */
/*         and it invokes EXSPUFI (supplying all parameters and with */
/*         "NEWAPPL(ssid)" ).                                        */
/*                                                                   */
/*      b) Command ('SPUFI' or 'SPUFI ssid') entered on the command  */
/*         line of any ISPF panel, where that command is defined as  */
/*         'SELECT CMD(%ESQL &ZPARM) NEWAPPL' in an ISPF command     */
/*         table.  Then ESQL invokes EXSPUFI with "NEWAPPL(ssid)"    */
/*         and the user would see a normal looking SPUFI primary     */
/*         panel (DSNESPØ1) and be able to use it in the normal way. */
/*                                                                   */
/*      c) If invoked (ONLINE) from another EXEC or CLIST:           */
/*         it allows you to run SQL without using the REXX-DB2 or    */
/*         DSNTEP2 programs, then you could browse the output or run */
/*         some output parsing routine to extract what you want.     */
```

```
/*         Here is some example pseudo code:                          */
/*            1). write the SQL into 'input.dsn' (RECFM=FB,LRECL=8Ø) */
/*            2). allocate file 'output.dsn'     (RECFM=FB or VB)    */
/*            3). SELECT CMD(%EXSPUFI SSID(ssid) INDSN('input.dsn')  */
/*                        OUTDSN('output.dsn') BR(NO)) NEWAPPL(ssid)  */
/*            4). parse file 'output.dsn' to check the SQLCODE       */
/*         It runs the SQL in 'input.dsn' and outputs to 'output.dsn'*/
/*         (but does not show the output).  Any parameters which are */
/*         not specified, will be defaulted in the DSNESPØ1 panel.   */
/*         *** Note that SPUFI will not run in TSO batch jobs.       */
/*-------------------------------------------------------------------*/
/* Externals:                                                        */
/*      DB2LIBS   REXX   .. allocate DB2 libraries                   */
/*      DSNESPØ1  Panel  .. modified version of IBM-supplied panel   */
/*      DSN       Pgm    .. standard IBM program to run SPUFI        */
/*-------------------------------------------------------------------*/
/* Written: 2ØØ3/Ø6/Ø5     Last Updated: 2ØØ3/11/26    by Ron Brown  */
/*===================================================================*/
   If Sysvar('SYSENV') = 'BACK' Then Do
      Say ''
      Say '*** EXSPUFI cannot run in TSO BATCH - abending RC=16'
      Say ''
      Exit 16
      End
   /*----------------------*/
   /* get start parameters */
   /*----------------------*/
   Arg parms
   Parse Upper Value ' 'parms With ,
      1 ' PANEL('  spufipnl ')',          /* show SPUFI panel?  */
      1 ' ALLOC(' lib_alloc ')',          /* DB2 library alloc  */
      1 ' SSID('   dsneovØ1 ')',          /* DB2 sub-system     */
      1 ' LOC('    spufiloc ')',          /* connect location   */
      1 ' INDSN('  spufidsn ')',          /* input dataset      */
      1 ' OUTDSN(' spufiout ')',          /* output dataset     */
      1 ' LRECL('  spufilen ')',          /* output LRECL       */
      1 ' BLK('    spufiblk ')',          /* output BLKSIZE     */
      1 ' RECFM('  spufifmt ')',          /* output RECFM       */
      1 ' UNIT('   spufiuni ')',          /* output UNIT        */
      1 ' BR('     spufibro ')',          /* browse output      */
      1 ' ISO('    spufiiso ')',          /* SPUFI isolation    */
      1 ' MAXNO('  spufimno ')',          /* max numeric field  */
      1 ' MAXCH('  spufimch ')',          /* max char field     */
      1 ' MAXSEL(' spufimax ')',          /* max SELECT lines   */
      1 ' COL('    spuficol ')',          /* column heading     */
      1 ' COM('    spuficom ')'           /* auto-commit        */
  /*-------------------------------------------------------*/
  /* Check parameters & pass them to SPUFI panel (DSNESPØ1) */
  /*-------------------------------------------------------*/
  If spufibro = '' Then
```

```
   spufibro = 'NO'
If spufiiso = '' Then
   spufiiso = 'CS'       /* 'CS' rather than default 'RR' */
If spufimax = '' Then
   spufimax = '2500'     /* 2500 rather than default 250  */
If spufimch = '' Then
   spufimch = '256'      /* 256 rather than default 80    */
If spuficom = '' Then
   spuficom = 'YES'
/*-- check what dataset name will be used for SPUFI output file --*/
 Address ISPEXEC                  /* commands go to ISPF services */
"VGET DSNESV16 PROFILE"
If spufiout <> '' Then Do
   If Right(spufiout,5) = ',COND' Then Do
      If DSNESV16 = '' Then Parse Var spufiout DSNESV16 ',COND'
      End
   Else
      DSNESV16 = spufiout
   End
/*-- prevent SPUFI from changing the DCB of a pre-allocated file --*/
If DSNESV16 <> '' & spufilen || spufiblk || spufifmt = '' Then Do
   If Listdsi(DSNESV16) = 0 Then Do
      spufilen = syslrecl
      spufiblk = sysblksize
      spufifmt = sysrecfm
      End
   End
/*-- pass the parameters to the (modified) SPUFI panel DSNESP01 --*/
"VPUT (DSNEOV01,SPUFILOC,SPUFIDSN,SPUFIOUT,SPUFIBRO,SPUFIISO,",
      "SPUFICOM,SPUFIMAX,SPUFILEN,SPUFIBLK,SPUFIFMT,SPUFIUNI,",
      "SPUFIPNL,SPUFICOL,SPUFIMNO,SPUFIMCH) PROFILE"
/*---------------------------------*/
/* DB2 library allocation for SPUFI */
/*---------------------------------*/
If lib_alloc <> 'NO' Then Do
   If lib_alloc = '' Then
      lib_alloc = 'DB2LIBS'          /* default allocation routine */
   "SELECT CMD(%"lib_alloc dsneov01")"
   End
/*-----------------------------------------------*/
/* Set various SPUFI defaults from DSNESP02 panel */
/*-----------------------------------------------*/
"VGET DSNESV1W PROFILE"   /* is this the first time for this ssid? */
If DSNESV1W <> 'SECOND TIME' Then Do  /* set missing SPUFI defaults*/
  /* standard except: max-sel-lines = 2500 & max-char-length = 256 */
   Parse Value '; 2500 4092 4096 VB SYSDA',
               '33 256 NAMES C SECOND TIME',
      With DSNESV2B DSNESV2D DSNESV2C DSNESV21 DSNESV22 DSNESV2E,
           DSNESV24 DSNESV25 DSNESV26 DSNESV3Z DSNESV1W
   "VPUT (DSNESV2B DSNESV2D DSNESV2C DSNESV21 DSNESV22 DSNESV2E",
```

```
          "DSNESV24 DSNESV25 DSNESV26 DSNESV3Z DSNESV1W) PROFILE"
     End
  /*-------------------------------------------------------------*/
  /* Invoke normal SPUFI directly                                */
  /* (alternatively you could invoke DSNESCØ1 CLIST to run SPUFI,  */
  /*  as follows:  "SELECT CMD(%DSNESCØ1 FUNC(SPUFI))"  )        */
  /*-------------------------------------------------------------*/
   Address TSO                        /* commands go to TSO */
  "NEWSTACK"                    /* start a new (empty) stack */
  Push ''
  Push 'END'
  Push 'SPUFI'
  "DSN SYSTEM("dsneovØ1") TEST(Ø) RETRY(Ø)"
  spufi_rc = rc
  "DELSTACK"                    /* finished with that stack */
  If spufi_rc > Ø Then
     Say '*** SPUFI ended with return code:' spufi_rc
  Return
```

DB2LIBS EXEC

Here is a sample EXEC to allocate the required DB2 libraries, which must be changed to meet your local standards. You probably have a similar allocation routine already. Note that for our purposes you need to concatenate a library containing the modified DSNESP01 panel before the standard IBM library.

An alternative approach could be to allocate your panel library to DDNAME = ISPPUSR, then it is automatically concatenated ahead of the libraries that are specified in the LIBDEF.

```
/*============================> REXX <================================*/
/* DB2LIBS: Allocate DB2 TSO/ISPF libraries for SPUFI               */
/*                                                    Version 2  */
/* Parameters Passed:                                               */
/* dsmbr - Data Sharing Member or Data Sharing Group Name           */
/*-----------------------------------------------------------------*/
/* Created: 2ØØ3/Ø6/Ø5     Last Updated: 2ØØ3/11/26    by Ron Brown  */
/*=================================================================*/
  Address ISPEXEC                    /* commands go to ISPF services */
  Parse Arg dsmbr .                      /* DB2 ssid or group       */
  If Right(dsmbr,1) = 'Ø' Then Do   /* If group - find local ssid    */
     smfid = MVSVAR('SYSSMFID')           /* MVS name (eg. S1Ø1,S1Ø2)*/
     DSMBR = Left(dsmbr,3)||Right(smfid,1)/* DB2 ssid (eg. SDB1,SDB2)*/
     End
  dsgrp = Left(dsmbr,3)||'Ø'        /* Data Sharing Group (eg SDBØ)  */
  "LIBDEF ISPLLIB DATASET ID('SYS1.DB2"dsgrp".SDSNEXIT'",
```

```
                          "'SYS1.DB2"dsgrp".SDSNLOAD'",
                          "'"dsgrp"."dsmbr".RUNLIB.LOAD')"
  "LIBDEF ISPMLIB DATASET ID('SYS1.DB2.SDSNSPFM') STACK"
  Call SET_PANELIBS
  "LIBDEF ISPPLIB DATASET ID("panelibs") STACK"
  /*-- this is only required if you use DSNESCØ1 clist ----*/
  /* Address TSO "ALTLIB ACT APPLICATION(CLIST) UNCOND ",  */
  /*             "DATASET('SYS1.DB2.NEW.SDSNCLST')"        */
  Return
  /*================================================================*/
  /* Make list of panel libraries, including one which contains a   */
  /* modified DSNESPØ1 panel to run SQL without displaying the panel.*/
  /*----------------------------------------------------------------*/
SET_PANELIBS:
  panelibs = "'SYS1.DB2.SDSNSPFP' 'SYS1.DB2.SDSNPFPE'"   /* IBM libs */
  panelibs = "'our.spufi.panelib'" panelibs
  Return
```

INSTALLATION

ESQLHELP panel should be copied to a library in your ISPPLIB concatenation.

DSNESP01 should be copied to a special library and modified as detailed above. Do not modify your SDSNPFPE SMP/E target library except via an SMP/E usermod!

ESQLJOB skeleton should be copied to a library in your ISPSLIB concatenation. Tailor it to suit your local JCL standards.

ESQL, EXSPUFI, and (optionally) DB2LIBS should be copied into a library in your SYSEXEC (or SYSPROC) concatenation.

Update ESQL to set the best default DB2 ssid, and any other defaults that you wish to change. Check the code for generating the name for SPUFI input and output datasets and amend it if necessary.

Update DB2LIBS to your local library naming standards, or alternatively update ESQL to invoke your own allocation routine instead of DB2LIBS.

SUMMARY

The improvements make this version much more flexible for

SPUFI use, adding the possibility of running the SQL in batch, and including a detailed HELP panel for the users.

Ron Brown (ron_brown@hotmail.com)
Principal Consultant
Triton Consulting (Germany)

Easy data load after table changes

If you want to change a column's attribute for an existing table, you have to drop the table and recreate it with new DDLs.

Before starting this process, you need to save rows in order to load them after recreating the table. Some problems can occur at load time while converting data types (eg decimal to char conversion).

The following REXX programs help to you to load data easily after a new table is created. The following conversion types are supported:

- CHAR TO VARCHAR,VARCHAR TO CHAR
- SMALLINT TO DECIMAL,DECIMAL TO SMALLINT
- INTEGER TO DECIMAL,DECIMAL TO INTEGER
- TIMESTAMP TO DATE
- INTEGER TO SMALLINT , SMALLINT TO INTEGER
- DEFAULT VALUE ASSIGNS FOR NEW COLUMNS
- SENSE CHANGED COLUMNS NAME
- SENSE DELETED COLUMNS

To run these programs:

- Execute the TABLS REXX program and release the TSAVE job. (This will copy to a temporary table rows within the table that is to be dropped.)

- After the TSAVE job completes successfully, drop and recreate the table with the new DDL.
- After recreating the table, run the TABLC REXX program and release the TCOPY job. (This will copy rows to the newly-created table with data conversion from the temporary table.)
- The working library is USERID.PD0.TABL. You can change it in the REXX programs.

The first program is the create job, which will create a new table just like the original one. After that it copies rows into the newly-created table from the original table using SQL statements like:

```
CREATE TABLE XCPY.TABLE_A LIKE XAAA.TABLE_A;
COMMIT;
INSERT INTO XCPY.TABLE_A SELECT * FROM XAAA.TABLE_A;
COMMIT;
```

The second program, after dropping and recreating the table that will be changed, creates a new job that will find all the required conversions and prepare SQL statements to load the new table like:

```
INSERT INTO XAAA.TABLE_A(COL1,COL2..) SELECT
      COL1,COL2,.. FROM XCPY.TABLE_A
```

Before running these programs, you have to create a tablespace for each DB2 subsystem. To create the tablespace, run the following SQL:

```
-- THIS DDL CREATES SAMPLE STOGRAGE GROUP , DATABASE AND TABLESPACE
-- IN THE GIVEN DB2 SUBSYSTEM.
-- X INDICATES SUBSYSTEM-ID
-- PLEASE REPLACE WITH D FOR DEVL DB2
--                     T FOR TEST DB2
--                     P FOR TEST DB2
-- PLEASE CHANGE VCATNAME WITH YOUR ALIAS
 CREATE STOGROUP XCPYSØ01
        VOLUMES ('*'      )
        VCAT  VCATNAME;

  CREATE DATABASE XCPYDØ01
         STOGROUP XCPYSØ01
         BUFFERPOOL BP2
```

```
        INDEXBP BP3  ;

CREATE TABLESPACE XCPYP001 IN XCPYD001
       USING STOGROUP XCPYS001
             PRIQTY  10000  SECQTY  10000
             FREEPAGE 0  PCTFREE 0
       GBPCACHE  CHANGED
       SEGSIZE    00032
       BUFFERPOOL BP2
       LOCKSIZE ANY
       CLOSE NO
                  ;
```

TABLP1 PANEL

```
)PANEL
)ATTR
# TYPE(TEXT)   INTENS(HIGH) HILITE(REVERSE)
+ TYPE(TEXT)   INTENS(HIGH) SKIP(ON)
_ TYPE(INPUT)  INTENS(LOW)  COLOR(YELLOW) PAD(_)
Q TYPE(OUTPUT) INTENS(HIGH)  COLOR(YELLOW)
$ TYPE(OUTPUT) INTENS(HIGH)  COLOR(YELLOW) HILITE(REVERSE)
)BODY
+
+          #  DATA LOAD AFTER THE TABLE CHANGE   +    User : Qwuser
+
+
+   DB2 subsystem id....   : _d+    D,T,V,E,V,W,P
+
+
+
+
+
+
+
+
+
+
+
+
+
+
+
+
+
+ MSG : $msg                                                      +
+     PF3 : RETURN                                  ENTER : PROCESS
)INIT
)proc
```

```
  IF (&wsys = 'TX')
      VER(&d,nonblank,list,D,T,E,V)
    IF (&wsys = 'PX')
      VER(&d,nonblank,list,P,W)

)END
```

TABLP2 PANEL

```
)PANEL
)ATTR
# TYPE(TEXT)   INTENS(HIGH) HILITE(REVERSE)
+ TYPE(TEXT)   INTENS(HIGH) SKIP(ON)
_ TYPE(INPUT)  INTENS(LOW)  COLOR(YELLOW) PAD(_)
Q TYPE(OUTPUT) INTENS(HIGH)  COLOR(YELLOW)
$ TYPE(OUTPUT) INTENS(HIGH)  COLOR(YELLOW) HILITE(REVERSE)
)BODY
+
+          #  DATA LOAD AFTER THE TABLE CHANGE   +    User : Qwuser
+
+
+   Db2 subsystem id....: Qd+    D,T,V,E,V,W,P
+
+   Owner of save process:Qau01    +
+
+        Table   Creator       Table name
+        ===============       ======================
+         Qac01    +           Qat01               +
+         Qac02    +           Qat02               +
+         Qac03    +           Qat03               +
+         Qac04    +           Qat04               +
+         Qac05    +           Qat05               +
+         Qac06    +           Qat06               +
+         Qac07    +           Qat07               +
+         Qac08    +           Qat08               +
+         Qac09    +           Qat09               +
+         Qac10    +           Qat10               +
+
+ MSG : $msg                                                    +
+     PF3 : RETURN                                  ENTER : PROCESS
)INIT
)proc
  IF (&wsys = 'TX')
      VER(&d,nonblank,list,D,T,E,V)
    IF (&wsys = 'PX')
      VER(&d,nonblank,list,P,W)
)END
```

TABLP3 PANEL

```
)PANEL
)ATTR
# TYPE(TEXT)   INTENS(HIGH) HILITE(REVERSE)
+ TYPE(TEXT)   INTENS(HIGH) SKIP(ON)
_ TYPE(INPUT)  INTENS(LOW)  COLOR(YELLOW) PAD(_)
Q TYPE(OUTPUT) INTENS(HIGH)  COLOR(YELLOW)
$ TYPE(OUTPUT) INTENS(HIGH)  COLOR(YELLOW) HILITE(REVERSE)
)BODY
+
+          #  DATA SAVE BEFORE THE TABLE CHANGE      +    User : Qwuser
+
+
+   Db2 subsystem id....: _d+    D,T,V,E,V,W,P
+
+
+      Table   Creator       Table name
+      ===============       ======================
+       _ac01    +           _at01               +
+       _ac02    +           _at02               +
+       _ac03    +           _at03               +
+       _ac04    +           _at04               +
+       _ac05    +           _at05               +
+       _ac06    +           _at06               +
+       _ac07    +           _at07               +
+       _ac08    +           _at08               +
+       _ac09    +           _at09               +
+       _ac1Ø    +           _at1Ø               +
+
+
+ MSG : $msg                                                      +
+     PF3 : RETURN                                  ENTER : PROCESS
)INIT
)proc
  IF (&wsys = 'TX')
      VER(&d,nonblank,list,D,T,E,V)
    IF (&wsys = 'PX')
      VER(&d,nonblank,list,P,W)
  )END
```

TABLS REXX PROGRAM

This REXX program prepares JCL to save data before the table is dropped.

```
/*                      rexx                                          */
/*********************************************************************/
/* This REXX program copies rows from table into the temporary table*/
```

```
/*********************************************************************/
 wuser = sysvar(sysuid)
 $ispexec libdef ispplib dataset id('$wuser$.pdØ.tabl')$
 wtar  = date('e')
 wsys = mvsvar('SYSNAME')
 wver = mvsvar('SYSOPSYS')

 $ispexec display panel(tablp3)$
 if rc ¬= Ø then return
 dbid = 'DB' || d || 'Ø'
 call create_table_list
 call jcl_stmt

  dd= Ø  ; cc = Ø

  do i = 1 to k

    dd= dd+ 1
    wcre = substr(ac.1,1,1) || 'CPY'
    wcre1 = substr(ac.1,1,1)
    dropddl.dd= ' DROP TABLE  ' || wcre || '.' || at.i || ';'
    dd= dd+ 1
    dropddl.dd= ' COMMIT; '
    cc = cc +1
    creddl.cc  = ' CREATE TABLE ' || wcre || '.' || at.i
    cc = cc +1
    creddl.cc  = '    LIKE ' || word(ac.i,1) || '.' || at.i
    cc = cc + 1
    creddl.cc  = '    IN ' || wcre1 || 'CPYDØØ1.' || wcre1 || 'CPYPØØ1;'
    cc  = cc  + 1
    creddl.cc  = ' COMMIT; '
    cc  = cc  +1
    creddl.cc  = '    '
    cc = cc +1
    creddl.cc = ' INSERT INTO ' || wcre || '.' || at.i
    cc   = cc  +1
    creddl.cc = ' SELECT * FROM ' ||word(ac.i,1)|| '.' ||at.i ||';';
    cc  = cc  +1
    creddl.cc  = '    '
  end
push ''
  push jcl.53
  do i = cc    to 1 by -1
     push creddl.i
  end
  do i = 52    to 41 by -1
     push jcl.i
  end
  push jcl.33
  do i = dd    to 1 by -1
```

```
      push dropddl.i
   end
   do i = 32    to 21 by -1
      push jcl.i
   end
   do i = 6     to 1  by -1
      push jcl.i
   end
     ww_mem = d || 'TSAVE'
     dsname2 = wuser || '.pdØ.tabl(' || ww_mem || ')'
     $alloc fi(savef) da($dsname2$) shr$
     $execio * diskw savef (finis$
     $free fi(savef)$
    $SUBMIT $wuser$.pdØ.tabl($ww_mem$)$
    say '*******************************************'
    say '* YOU SHOULD RELEASE TSAVE JOB FROM SDSF.   *'
    say '*******************************************'
   push ''
   do i = k to 1  by -1
      line1 = wuser || ' ' || ac.i || ' ' ||  at.i
      push line1
   end
     ww_mem = d || 'TLIST'
     dsname2 = wuser || '.pdØ.tabl(' || ww_mem || ')'
     $alloc fi(savef) da($dsname2$) shr$
     $execio * diskw savef (finis$
     $free fi(savef)$
   exit
 create_table_list:
           ac.1     = acØ1 ; at.1     = atØ1
           ac.2     = acØ2 ; at.2     = atØ2
           ac.3     = acØ3 ; at.3     = atØ3
           ac.4     = acØ4 ; at.4     = atØ4
           ac.5     = acØ5 ; at.5     = atØ5
           ac.6     = acØ6 ; at.6     = atØ6
           ac.7     = acØ7 ; at.7     = atØ7
           ac.8     = acØ8 ; at.8     = atØ8
           ac.9     = acØ9 ; at.9     = atØ9
           ac.1Ø    = ac1Ø ; at.1Ø    = at1Ø
      k = Ø
      do i = 1 to 6
         if ac.i    > ' ' & at.i > ' ' then do
                 k = k + 1
                 ac1.k    = ac.i
                 at1.k    = at.i
         end
      end
return

 jcl_stmt:
```

```
   jcl.1  = '//TSAVE  JOB ,,CLASS=A,MSGCLASS=X,                    '
   jcl.2  = '//         MSGLEVEL=(1,1),TYPRUN=HOLD,REGION=ØM        '
   jcl.3  = '//*                                                    '
   jcl.4  = '//JOBLIB   DD DISP=SHR,DSN=' || d || 'DSN.SDSNEXIT     '
   jcl.5  = '//         DD DISP=SHR,DSN=' || d || 'DSN.SDSNLOAD     '
   jcl.6  = '//         DD DISP=SHR,DSN=' || d || 'DSN.RUNLIB.LOAD  '
   jcl.21 = '//*************************************************'
   jcl.22 = '//*        DROP EXISTING TABLES WITH QUAL XCPY        '
   jcl.23 = '//*************************************************'
   jcl.24 = '//*'
   jcl.25 = '//DROPTB    EXEC PGM=IKJEFTØ1,REGION=512K         '
   jcl.26 = '//SYSPRINT  DD SYSOUT=*                           '
   jcl.27 = '//SYSTSPRT  DD SYSOUT=*                           '
   jcl.28 = '//SYSTSIN   DD *                                  '
   jcl.29 = '  DSN SYSTEM(' || dbid || ')                       '
   jcl.3Ø = '       RUN PROGRAM(DSNTEP2) PLAN(DSNTEP2)         '
   jcl.31 = '  END                                             '
   jcl.32 = '//SYSIN    DD *                                   '
   jcl.33 = '/*'
   jcl.41 = '//*************************************************'
   jcl.42 = '//*   CREATE TABLE WITH QUAL XCPY AND COPY ROWS    '
   jcl.43 = '//*************************************************'
   jcl.44 = '//*'
   jcl.45 = '//SAVETB    EXEC PGM=IKJEFTØ1,REGION=512K         '
   jcl.46 = '//SYSPRINT  DD SYSOUT=*                           '
   jcl.47 = '//SYSTSPRT  DD SYSOUT=*                           '
   jcl.48 = '//SYSTSIN   DD *                                  '
   jcl.49 = '  DSN SYSTEM(' || dbid || ')                       '
   jcl.5Ø = '       RUN PROGRAM(DSNTEP2) PLAN(DSNTEP2)         '
   jcl.51 = '  END                                             '
   jcl.52 = '//SYSIN    DD *                                   '
   jcl.53 = '/*'
return
```

TABLC REXX PROGRAM

This REXX program prepares JCL to load data into the new table from the temporary table.

```
/*                     rexx                                     */
 /*************************************************************/
 /* This REXX program copies rows from temporary table to      */
 /* original table that was newly created.                     */
 /*************************************************************/
  wuser = sysvar(sysuid)
  $ispexec libdef ispplib dataset id('$wuser$.pdØ.tabl')$
  wtar  = date('e')
  wsys = mvsvar('SYSNAME')
  wver = mvsvar('SYSOPSYS')
```

```
 $ispexec display panel(tablp1)$
 if rc ¬= Ø then return
 call table_list_read
 $ispexec display panel(tablp2)$
 if rc ¬= Ø then return
 if s_flag = '1' then do
        say 'SAVE PROCESS WAS DONE WITH DIFFERENT USER-ID  '
        say '    YOU CANNOT CONTINUE THE PROCESS.'
    return
 end
 dbid = 'DB' || d || 'Ø'
 ic   = Ø
 k    = Ø
 do  mems = 1 to t
      cpys = Ø
      hsts = Ø
      wtcre = word(ac.mems,1)
      wttab = word(at.mems,1)
      wctab = word(at.mems,1)
      wccre = substr(ac.mems,1,1) || 'CPY'
      call select_hst
      call select_cpy
      call copy_prepare
 call jcl_stmt
 push ''
 push jcl.33
 do k = ic    to 1 by -1
    push instmt.k
 end
 do k = 32  to 2Ø by -1
    push jcl.k
 end
 do k = 6   to 1 by -1
    push jcl.k
 end
   ww_mem = d || 'tcopy'
   dsname1 = wuser || '.pdØ.tabl(' || ww_mem || ')'
   $alloc fi(cpy) da($dsname1$) shr$
   $execio * diskw cpy  (finis$
   $free fi(cpy)$
 $SUBMIT $wuser$.pdØ.tabl($ww_mem$)$
 say '*********************************************'
 say '*  YOU SHOULD RELEASE TCOPY JOB FROM SDSF.   *'
 say '*********************************************'
exit
/*************************************************************/
 /* preparing SQL statements for target table to capture     */
 /* catalog information.                                      */
 /*************************************************************/
select_hst:
```

```
    'subcom dsnrexx'
    if rc then
    s_rc = rxsubcom('ADD','DSNREXX','DSNREXX')
    address dsnrexx
    connect dbid
    if rc ¬= Ø then return
  sqlstmt = ,
   ' select tbname,digits(colno),',
   ' substr(name,1,18),substr(coltype,1,8),digits(length)',
   ' from sysibm.syscolumns',
   ' where tbcreator = ' || '''' || wtcre || '''' || ' and ',
   '       tbname = ' || '''' || wttab || '''',
   ' order by 2                '
  'execsql declare c1 cursor for s1'
  if sqlcode ¬= Ø then  call sql_error
  'execsql prepare s1 into :outsqlda from :sqlstmt'
  if sqlcode ¬= Ø then  call sql_error
  'execsql open C1'
  if sqlcode ¬= Ø then  call sql_error
  'execsql fetch c1 using descriptor :outsqlda'
  if sqlcode < Ø then  call sql_error
    do while(sqlcode = Ø)
           hsts = hsts + 1
           hcolno.hsts   =   word(outsqlda.2.sqldata,1)
           hcolname.hsts =   word(outsqlda.3.sqldata,1)
           hcoltype.hsts =   word(outsqlda.4.sqldata,1)
           hcollen.hsts  =   word(outsqlda.5.sqldata,1)
       'execsql fetch c1 using descriptor :outsqlda'
    end
    if sqlcode < Ø then  call sql_error
    'execsql close c1 '
    if sqlcode < Ø then  call sql_error
    'disconnect'
    if sqlcode < Ø then  call sql_error
    s_rc = rxsubcom('DELETE','DSNREXX','DSNREXX')
 return;
select_cpy:
    'subcom dsnrexx'
    if rc then
    s_rc = rxsubcom('ADD','DSNREXX','DSNREXX')
    address dsnrexx
    connect dbid
    if rc ¬= Ø then return
  sqlstmt = ,
   ' select tbname,digits(colno),',
   ' substr(name,1,18),substr(coltype,1,8),digits(length)',
   ' from sysibm.syscolumns',
   ' where tbcreator = ' || '''' || wccre || '''' || ' and ',
   '       tbname = ' || '''' || wctab || '''',
   ' order by 2                '
```

```
  'execsql declare c1 cursor for s1'
  if sqlcode ¬= Ø then  call sql_error
  'execsql prepare s1 into :outsqlda from :sqlstmt'
  if sqlcode ¬= Ø then  call sql_error
  'execsql open C1'
  if sqlcode ¬= Ø then  call sql_error
  'execsql fetch c1 using descriptor :outsqlda'
  if sqlcode < Ø then  call sql_error
    do while(sqlcode = Ø)
           cpys = cpys + 1
           ccolno.cpys   =   word(outsqlda.2.sqldata,1)
           ccolname.cpys =   word(outsqlda.3.sqldata,1)
           ccoltype.cpys =   word(outsqlda.4.sqldata,1)
           ccollen.cpys  =   word(outsqlda.5.sqldata,1)
       'execsql fetch c1 using descriptor :outsqlda'
    end
    if sqlcode < Ø then  call sql_error
    'execsql close c1 '
    if sqlcode < Ø then  call sql_error
    'disconnect'
    if sqlcode < Ø then  call sql_error
    s_rc = rxsubcom('DELETE','DSNREXX','DSNREXX')
 return;
/*************************************************************/
/*  Preparing SQL statements to copy rows.                   */
/*************************************************************/
copy_prepare:
     ic = ic + 1
     instmt.ic = 'INSERT INTO ' || wtcre || '.' || wttab
     ic = ic + 1
     instmt.ic = '('
     do hh = 1 to hsts
        ic = ic + 1
        if hh < hsts  then
              instmt.ic = '       '  || hcolname.hh || ','
           else
              instmt.ic = '       '  || hcolname.hh  || ')'
      end
     ic = ic + 1
     instmt.ic = 'SELECT '
     do hh = 1 to hsts
       ww = 'Ø'
       do ss = 1 to cpys
                 /************************************************/
                 /*  if column name types are same               */
                 /************************************************/
          if hcolname.hh = ccolname.ss then
            if hcoltype.hh = ccoltype.ss then
               do
                 ic = ic + 1
```

```
        instmt.ic = '       ' || ccolname.ss || ','
        ww = '1'
      end
        /*************************************************/
        /*  Column names are same but column types are    */
        /*  different.  varchar ----> char               */
        /*************************************************/
if hcolname.hh = ccolname.ss  then
if hcoltype.hh ¬= ccoltype.ss then
if hcoltype.hh = 'CHAR'       then
if ccoltype.hh = 'VARCHAR'    then
    do
      ic = ic + 1
      instmt.ic = '       ' || ccolname.ss || ','
      ww = '1'
    end
        /*************************************************/
        /*  Column names are same but column types are    */
        /*  different.  char----> varchar                */
        /*************************************************/
if hcolname.hh = ccolname.ss  then
if hcoltype.hh ¬= ccoltype.ss then
if hcoltype.hh = 'VARCHAR'    then
if ccoltype.hh = 'CHAR'       then
    do
      ic = ic + 1
      instmt.ic = '       ' || ccolname.ss || ','
      ww = '1'
    end
        /*************************************************/
        /*  Column names are same but column types are    */
        /*  different.  decimal--> smallint              */
        /*************************************************/
if hcolname.hh = ccolname.ss  then
if hcoltype.hh ¬= ccoltype.ss then
if hcoltype.hh = 'SMALLINT'   then
if ccoltype.hh = 'DECIMAL'    then
    do
      ic = ic + 1
      instmt.ic = '       ' || ccolname.ss || ','
      ww = '1'
    end
        /*************************************************/
        /*  Column names are same but columns types are   */
        /*  different.  smallint--> decimal              */
        /*************************************************/
if hcolname.hh = ccolname.ss  then
if hcoltype.hh ¬= ccoltype.ss then
if hcoltype.hh = 'DECIMAL'    then
if ccoltype.hh = 'SMALLINT'   then
```

```
    do
      ic = ic + 1
      instmt.ic = '       ' || ccolname.ss || ','
      ww = '1'
    end
       /************************************************/
       /*  Column names are same but column types are  */
       /*  different.  smallint--> integer             */
       /************************************************/
if hcolname.hh = ccolname.ss  then
if hcoltype.hh ¬= ccoltype.ss then
if hcoltype.hh = 'INTEGER'    then
if ccoltype.hh = 'SMALLINT'   then
    do
      ic = ic + 1
      instmt.ic = '       ' || ccolname.ss || ','
      ww = '1'
    end
       /************************************************/
       /*  Column names are same but column types are  */
       /*  different.  integer--> smallint             */
       /************************************************/
if hcolname.hh = ccolname.ss  then
if hcoltype.hh ¬= ccoltype.ss then
if hcoltype.hh = 'SMALLINT'   then
if ccoltype.hh = 'INTEGER'    then
    do
      ic = ic + 1
      instmt.ic = '       ' || ccolname.ss || ','
      ww = '1'
    end
       /************************************************/
       /*  Column names are same but column types are  */
       /*  different.  integer--> decimal              */
       /************************************************/
if hcolname.hh = ccolname.ss  then
if hcoltype.hh ¬= ccoltype.ss then
if hcoltype.hh = 'DECIMAL'    then
if ccoltype.hh = 'INTEGER'    then
    do
      ic = ic + 1
      instmt.ic = '       ' || ccolname.ss || ','
      ww = '1'
    end
       /************************************************/
       /*  Column names are same but column types are  */
       /*  different.  decimal--> integer              */
       /************************************************/
if hcolname.hh = ccolname.ss  then
if hcoltype.hh ¬= ccoltype.ss then
```

```
if hcoltype.hh = 'INTEGER'    then
if ccoltype.hh = 'DECIMAL'    then
    do
      ic = ic + 1
      instmt.ic = '       ' || ccolname.ss || ','
      ww = '1'
    end
       /************************************************/
       /*  Column names are same but column types are    */
       /*  different.  timestamp--> date                 */
       /************************************************/
if hcolname.hh = ccolname.ss  then
if hcoltype.hh ¬= ccoltype.ss then
if hcoltype.hh = 'DATE'       then
if ccoltype.hh = 'TIMESTMP'   then
    do
      ic = ic + 1
      instmt.ic = '  DATE(' || ccolname.ss || '),'
      ww = '1'
    end
       /************************************************/
       /*  Column names are same but column types are    */
       /*  different.  smallint--> char                  */
       /************************************************/
if hcolname.hh = ccolname.ss  then
if hcoltype.hh ¬= ccoltype.ss then
if hcoltype.hh = 'CHAR'       then
if ccoltype.hh = 'SMALLINT'   then
    do
      ic = ic + 1
      instmt.ic = 'DIGITS(' || ccolname.ss || '),'
      ww = '1'
    end
       /************************************************/
       /*  Column names are same but column types are    */
       /*  different.  integer--> char                   */
       /************************************************/
if hcolname.hh = ccolname.ss  then
if hcoltype.hh ¬= ccoltype.ss then
if hcoltype.hh = 'CHAR'       then
if ccoltype.hh = 'INTEGER'    then
    do
      ic = ic + 1
      instmt.ic = 'DIGITS(' || ccolname.ss  || '),'
      ww = '1'
    end
       /************************************************/
       /*  Column names are same but column types are    */
       /*  different.  decimal--> char                   */
       /************************************************/
```

```
   if hcolname.hh = ccolname.ss  then
   if hcoltype.hh ¬= ccoltype.ss then
   if hcoltype.hh = 'CHAR'       then
   if ccoltype.hh = 'DECIMAL'    then
       do
         ic = ic + 1
         instmt.ic = 'DIGITS(' || ccolname.ss  || '),'
         ww = '1'
       end
end
         /*************************************************/
         /*  if column names are different (was changed)   */
         /*************************************************/
if ww = 'Ø' then
  do ss = 1 to cpys
    if hcolname.hh ¬= ccolname.ss  then
    if hcoltype.hh = ccoltype.ss   then
    if hcollen.hh = ccollen.ss     then
    if hcolno.hh = ccolno.ss       then
       do
         ic = ic + 1
         instmt.ic = '       ' || ccolname.ss || ','
         ww = '1'
       end
  end
         /*************************************************/
         /*  if column is a new column                    */
         /*  default value is given.                      */
         /*************************************************/
  if ww = 'Ø' then
     do
       ic = ic + 1
       lenx = length(hcoltype.hh)
       if hcoltype.hh = 'CHAR' then
         instmt.ic = '       '' '','
       if hcoltype.hh = 'VARCHAR' then
         instmt.ic = '       '' '','
       if hcoltype.hh = 'DECIMAL' then
         instmt.ic = '       Ø,'
       if hcoltype.hh = 'INTEGER' then
         instmt.ic = '       Ø,'
       if hcoltype.hh = 'SMALLINT' then
         instmt.ic = '       Ø,'
       if hcoltype.hh = 'DATE' then
         instmt.ic = 'CURRENT DATE,'
       if hcoltype.hh = 'TIME' then
         instmt.ic = 'CURRENT TIME,'
       if hcoltype.hh = 'TIMESTMP' then
         instmt.ic = 'CURRENT TIMESTAMP,'
     end
```

```
      end
         lenx = length(instmt.ic)
         instmt.ic = substr(instmt.ic,1,lenx-1)

         ic = ic + 1
         instmt.ic = '  FROM ' || wccre || '.' || wctab || ';'
         ic = ic + 1
         instmt.ic = 'COMMIT;'
  return
/*********************************************************************/
/* table list read                                                   */
/*********************************************************************/
table_list_read:
     ww_mem = d || 'TLIST'
    $alloc fi(tblist) da($wuser$.pdØ.tabl($ww_mem$)) shr$
    $execio * diskr tblist (stem tbnameØ.$
    if rc ¬= Ø then
       do
           say ' dataset cannot read.'
           return
       end
    $execio Ø diskr tblist (finis$
    if rc ¬= Ø then
       do
           say ' dataset cannot read.'
           return
       end
     $free fi(tblist)$
 t = tbnameØ.Ø
 if t > Ø then
    do
      auØ1 = word(tbnameØ.1,1)
      do i = 1 to 1Ø
         ac.i = ' '
         at.i = ' '
      end
      do i = 1 to t
         ac.i = word(tbnameØ.i,2)
         at.i = word(tbnameØ.i,3)
      end
  end
  s_flag = 'Ø'
  if wuser ¬= auØ1 then do
      msg = 'SAVE PROCESS WAS DONE WITH DIFFERENT USER'
      s_flag = '1'
  end
     acØ1 =  ac.1  ; atØ1 = at.1
     acØ2 =  ac.2  ; atØ2 = at.2
     acØ3 =  ac.3  ; atØ3 = at.3
     acØ4 =  ac.4  ; atØ4 = at.4
```

```
     ac05 =  ac.5  ; at05 = at.5
     ac06 =  ac.6  ; at06 = at.6
     ac07 =  ac.7  ; at07 = at.7
     ac08 =  ac.8  ; at08 = at.8
     ac09 =  ac.9  ; at09 = at.9
     ac10 =  ac.10 ; at10 = at.10
return

 jcl_stmt:
   jcl.1  = '//TCOPY    JOB ,,CLASS=A,MSGCLASS=X,                    '
   jcl.2  = '//         MSGLEVEL=(1,1),TYPRUN=HOLD,REGION=0M         '
   jcl.3  = '//*                                                     '
   jcl.4  = '//JOBLIB   DD DISP=SHR,DSN=' || d || 'DSN.SDSNEXIT      '
   jcl.5  = '//         DD DISP=SHR,DSN=' || d || 'DSN.SDSNLOAD      '
   jcl.6  = '//         DD DISP=SHR,DSN=' || d || 'DSN.RUNLIB.LOAD   '
   jcl.20 = '//*************************************************'
   jcl.21 = '//*  COPY ROWS FROM TABLE WITH QUAL XCPY INTO          '
   jcl.22 = '//*    NEWLY CREATED TABLE                            '
   jcl.23 = '//*************************************************'
   jcl.24 = '//*'
   jcl.25 = '//COPYTB    EXEC PGM=IKJEFT01,REGION=512K         '
   jcl.26 = '//SYSPRINT  DD SYSOUT=*                           '
   jcl.27 = '//SYSTSPRT  DD SYSOUT=*                           '
   jcl.28 = '//SYSTSIN   DD *                                  '
   jcl.29 = '  DSN SYSTEM(' || dbid || ')                       '
   jcl.30 = '       RUN PROGRAM(DSNTEP2) PLAN(DSNTEP2)         '
   jcl.31 = '  END                                             '
   jcl.32 = '//SYSIN    DD *                                   '
   jcl.33 = '/*'
return
 /*************************************************************/
 /*    sql_error                                              */
 /*************************************************************/
 sql_error:
      say 'SQL error.............'
      say sqlcode
      say sqlstmt
      'execsql close c1 '
      exit
```

EXAMPLES

DDLs of table that will be changed:

```
CREATE TABLE DDBA.TABLE_X
          (
           CMP_CODE            SMALLINT  NOT NULL ,
           OFFER_NUM           INTEGER   NOT NULL ,
           OFFER_ST_CODE       CHAR(1)   NOT NULL  WITH DEFAULT,
```

```
          PRD_TYPE_CODE       SMALLINT  NOT NULL  WITH DEFAULT,
          PRD_ORDER_NUM1      DEC(15,2) NOT NULL  WITH DEFAULT,
          PRD_ORDER_NUM2      INTEGER   NOT NULL  WITH DEFAULT,
          ENT_DATE            TIMESTAMP NOT NULL  WITH DEFAULT,
          ENT_OP_ID           CHAR(8)   NOT NULL  WITH DEFAULT,
          OFFER_NAME          CHAR(100) NOT NULL  WITH DEFAULT)
IN DDBAD001.DDBAP100;
NEW DDL'S OF TABLE :
CREATE TABLE DDBA.TABLE_X
         (
          CMP_CODE            INTEGER      NOT NULL ,
          OFFER_NUM           SMALLINT     NOT NULL ,
          OFFER_ST_CODE       CHAR(1)      NOT NULL  WITH DEFAULT,
          PRD_TYPE_CODE       CHAR(5)      NOT NULL  WITH DEFAULT,
          PRD_ORDER_NUM1      INTEGER      NOT NULL  WITH DEFAULT,
          PRD_ORDER_NUM2      DEC(15,0)    NOT NULL  WITH DEFAULT,
          ENT_DATE            DATE         NOT NULL  WITH DEFAULT,
          ENT_OP_ID           CHAR(10)     NOT NULL  WITH DEFAULT,
          OFFER_NAME          VARCHAR(100) NOT NULL  WITH DEFAULT,
          UPD_DATE            DATE         NOT NULL  WITH DEFAULT,
          UPD_OP_ID           CHAR(10)     NOT NULL  WITH DEFAULT)
IN DDBAD001.DDBAP100;
```

JCL prepared by TABLS:

```
//TSAVE  JOB ,,CLASS=A,MSGCLASS=X,
//         MSGLEVEL=(1,1),TYPRUN=HOLD,REGION=0M
//*
//JOBLIB   DD DISP=SHR,DSN=DDSN.SDSNEXIT
//         DD DISP=SHR,DSN=DDSN.SDSNLOAD
//         DD DISP=SHR,DSN=DDSN.RUNLIB.LOAD
//*****************************************************
//*        DROP EXISTING TABLES WITH QUAL XCPY
//*****************************************************
//*
//DROPTB    EXEC PGM=IKJEFT01,REGION=512K
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD *
  DSN SYSTEM(DBD0)
       RUN PROGRAM(DSNTEP2) PLAN(DSNTEP2)
  END
//SYSIN    DD *
 DROP TABLE  DCPY.TABLE_X;
 COMMIT;
/*
//*****************************************************
//*   CREATE TABLE WITH QUAL XCPY AND COPY ROWS
//*****************************************************
//*
```

```
//SAVETB    EXEC PGM=IKJEFTØ1,REGION=512K
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD *
  DSN SYSTEM(DBDØ)
       RUN PROGRAM(DSNTEP2) PLAN(DSNTEP2)
  END
//SYSIN    DD *
 CREATE TABLE DCPY.TABLE_X
    LIKE DDBA.TABLE_X
    IN DCPYDØØ1.DCPYPØØ1;
 COMMIT;
 INSERT INTO DCPY.TABLE_X
 SELECT * FROM DDBA.TABLE_X;
/*
```

JCL prepared by TABLC:

```
//TCOPY    JOB ,,CLASS=A,MSGCLASS=X,
//         MSGLEVEL=(1,1),TYPRUN=HOLD,REGION=ØM
//*
//JOBLIB   DD DISP=SHR,DSN=DDSN.SDSNEXIT
//         DD DISP=SHR,DSN=DDSN.SDSNLOAD
//         DD DISP=SHR,DSN=DDSN.RUNLIB.LOAD
//*****************************************************
//*  COPY ROWS FROM TABLE WITH QUAL XCPY TO
//*    NEWLY CREATED TABLE
//*****************************************************
//*
//COPYTB    EXEC PGM=IKJEFTØ1,REGION=512K
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD *
  DSN SYSTEM(DBDØ)
       RUN PROGRAM(DSNTEP2) PLAN(DSNTEP2)
  END
//SYSIN    DD *
INSERT INTO DDBA.TABLE_X
(
       CMP_CODE,
       OFFER_NUM,
       OFFER_ST_CODE,
       PRD_TYPE_CODE,
       PRD_ORDER_NUM1,
       PRD_ORDER_NUM2,
       ENT_DATE,
       ENT_OP_ID,
       OFFER_NAME,
       UPD_DATE,
       UPD_OP_ID)
```

```
SELECT
       CMP_CODE,
       OFFER_NUM,
       OFFER_ST_CODE,
DIGITS(PRD_TYPE_CODE),
       PRD_ORDER_NUM1,
       PRD_ORDER_NUM2,
  DATE(ENT_DATE),
       ENT_OP_ID,
       OFFER_NAME,
CURRENT DATE,
       ' '
  FROM DCPY.TABLE_X;
COMMIT;
/*
```

Ali Ozturk
Database Administrator
Pamukbank (Turkey)

IBM has announced general availability of DB2 Universal database for z/OS Version 8 for IBM's eServer zSeries mainframes.

The new database software delivers over 100 new features and functions that help businesses automate the way information flows throughout the enterprise. Additionally, new autonomic database features boost system performance and availability while easing application porting across multiple platforms, says IBM.

DB2 for z/OS Version 8 software simplifies application development with Java, enabling businesses to deploy new applications from a distributed environment to the mainframe using advanced SQL functionality that masks the complexity of deployment across platforms.

DB2 for z/OS Version 8 customers can now access and integrate business information in both distributed and mainframe DB2 data stores, leveraging their existing DB2 database skill set while extending the value of new and existing applications across all platforms.

Autonomic features include point-in-time recovery functionality, and there's an enhanced Query Management Facility (QMF) business intelligence tool.

A new tool for WebSphere users lets any Web browser become a zero-maintenance thin client for visual access to DB2 data.

For further information contact your local IBM representative.

* * *

QED Business Systems has announced the XML Extender options of FireXML for DB2.

The FireXML XML Extender options allow XML formatted data to be added directly to mainframe DBMSs, and for XML formatted files to be created directly from the data in mainframe DBMSs. XML has

With the XML Extender options of the FireXML suite, mainframe DBMSs can now communicate via MQSeries, EDI, ebXML, FTP, and sendmail.

Using the mapping facility of the FireXML XML Extenders, XML data structures are mapped directly to the fields of mainframe DBMS tables. Then by using the XML Extenders in standard batch processing, XML formatted data is added directly to, or extracted from, mainframe DBMS tables. All parsing takes place on the mainframe.

For further information contact:
QED Business Systems, QED House, 430 Bath Road, Slough SL1 6BB, UK.
Tel: (01628) 660025.
URL: http://www.qedbusiness.com/guard_download.asp?t=FireXMLDB2.

* * *

Informatica has announced a partnership deal with IBM so it can leverage Informatica's data integration software when migrating new customers to DB2 UDB. Using the automated Informatica solution will help IBM fuel new DB2 UDB sales, they claim.

Informatica provides a migration solution helping customers migrate from competitive database platforms to DB2 UDB. The migration is an automated engine-driven process, helping users avoid SQL scripting.

For further information contact:
Informatica, 2100 Seaport Boulevard, Redwood City, CA 94063, USA.
Tel: (650) 385 5000.
URL: http://www.informatica.com/News/Press+Releases/2004/03152004a_ibm.htm.