77

DB2 update

March 1999

In this issue

DB2 Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
E-mail: xephon@compuserve.com

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75077-2150
USA
Telephone: 940 455 7050

Contributions
Articles published in *DB2 Update* are paid for at the rate of £170 ($250) per 1000 words and £90 ($140) per 100 lines of code for original material. 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*.

DB2 Update on-line
Code from *DB2 Update* can be downloaded from our Web site at http://www.xephon.com; you will need the user-id shown on your address label.

Editor
Robert Burgess

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 1994 issue, are available separately to subscribers for £22.50 ($33.50) each including postage.

Displaying DB2 active log status

INTRODUCTION

DB2 active log status monitoring is a regular activity in many installations. Monitoring consists of being aware of log status, checking whether two copies of a log are synchronized, and also checking whether full logs were archived and, if not, how full the logs are. This monitoring is crucial, especially if there is going to be considerable batch activity with a lot of insert/update/delete activity (usually during off-peak hours). If there is a problem with the off-loading of full logs, and a batch program with many changes is running, the logs can become full and the DB2 subsystem will abend. In many installations, monitoring is often done by operators because database administrators are not present at the time.

The information required is recorded in DB2 subsystem Bootstrap Datasets (BSDS). However, there is lot of information in BSDSs that need not be presented to operators during the simple monitoring described (information about archive log datasets, conditional restart records, system checkpoints, etc).

All information recorded in BSDSs can be extracted into formatted output by using the DB2 utility DSNJU004. The operator must log-in to TSO, submit the job that executes the utility, list the job output, and scan through a lot of information to find the part that shows the status of the logs and how full they are.

There are also some statuses that will not mean anything to the operator (only to a DBA), and by looking at values for the highest RBA written, the highest RBA offloaded, and the RBA ranges of each active log, it is difficult to see how full the logs are.

THE PROBLEM

The problem is to present the operator on the JES console with the status of DB2 active logs, and all other necessary information, in an easily understandable way.

THE SOLUTION

As a solution, the operator submits a job by calling it from the JES console. This will give the operator output similar to that shown in Figure 1.

In normal circumstances, the output for both copies of the active log is the same. If the copies are not the same, they are out of synchronization, and appropriate action should be taken by the DBA.

The DBA can show the operators what the output would look like for critical situations in the DB2 subsystem, and how these situations can be recognized (for example, all logs are 100% full except for one; log status is stopped or truncated; different full percentage or status for active log copy 1 and 2 datasets, etc), and what the procedures are for these critical situations.

The DBA can also customize the REXX EXEC to change the status into more easly understandable words, ie REUSABLE into EMPTY, STOPPED into CALL IMM. 22031, NOTREUSABLE into CURRENT, etc.

```
DISPLAY DB2 ACTIVE LOG STATUS. DB2 SSID: DSNP
LOG MAP OF BOOTSTRAP DATASET DSNP31Ø.BSDSØ1

     ACTIVE LOG COPY 1 ...

     DS#       FULL%     STATUS
      1          Ø,ØØ    REUSABLE
      2        1ØØ,ØØ    STOPPED
      3         73,15    NOTREUSABLE

     ACTIVE LOG COPY 2 ...

     DS#       FULL%     STATUS
      1          Ø,ØØ    REUSABLE
      2        1ØØ,ØØ    STOPPED
      3         73,15    NOTREUSABLE
```

Figure 1: Example of output

Further information about statuses can be obtained from the appropriate DB2 manuals.

DACTSSID

The job DACTSSID should be saved in the PDS accessible to the disk reader. For each DB2 subsystem there should be a separate job – the name of the job should contain the DB2 SSID, so they can be easily distinguished.

Save the job into the appropriate PDS under an appropriate name (ie DACTDSN, DACTDSNP) and change all occurrences of the word SSID into the name of the DB2 subsystem-id.

The following job should be customized to suit your environment:

```
//DACTSSID  JOB 'RAØ1IVA','RAØ1IVA',
//          CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),
//          NOTIFY=RAØ1IVA
//***************************************************************
//* DISPLAYING ON CONSOLE DB2 ACTIVE LOG STATUS
//***************************************************************
//* SPECIFY THE DATASET NAMES ACCORDING TO YOUR INSTALLATION
//*   JOBLIB     CREATED DURING DB2 INSTALLATION
//*   SYSUT1     BOOTSTRAP DATASET FOR THE DB2 SUBSYSTEM SSID
//*   SYSEXEC    PDS WHERE THE EXEC RESIDES
//*
//*   CHANGE THE SSID WITH THE CORRECT DB2 SUBSYSTEM ID
//*          FOR WHICH THE JOB WILL BE EXECUTED
//*   FOR EACH DB2 SUBSYSTEM CREATE SEPARATE JOB
//*
//*   SAVE THIS JOB IN THE PDS WHERE IT CAN BE CALLED FROM THE
//*         DISK READER FROM THE JES CONSOLE
//*
//JOBLIB   DD  DSN=DSN31Ø.SDSNLOAD,DISP=SHR
//DSNJUØØ4 EXEC PGM=DSNJUØØ4
//SYSUT1   DD DSN=DSNT31Ø.BSDSØ1,DISP=SHR
//SYSPRINT DD DSN=&BSDSOUT,DISP=(NEW,PASS),
//            UNIT=SYSDA,SPACE=(TRK,(1,1),RLSE),
//            DCB=(RECFM=FB,LRECL=133,BLKSIZE=3Ø59)
//DACTSSID  EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø
//SYSEXEC  DD DISP=SHR,DSN=PRD1ESA.DB2.FIC.EXEC
//SYSTSPRT DD SYSOUT=*
//BSDSIN   DD DSN=&BSDSOUT,DISP=(OLD,DELETE)
//SYSTSIN  DD *
%DB2ACTLG
SSID
/*
```

DB2ACTLG

Save the REXX EXEC into PDS as noted by the SYSEXEC DD statement of the DACTSSID job given above. The EXEC will use the SSID parameter given in the SYSTSIN DD statement of the DACTSSID job.

Customize the PRF variable (BSDS dataset name prefix) to suit your installation. Here is the listing:

```
/* REXX *************************************************************/
/*******************************************************************/
/*   EXEC : DB2ACTLG                                               */
/*                                                                 */
/*   EXEC for Displaying on Console DB2 Active Log status          */
/*                                                                 */
/*   Authority: RACF                                               */
/*                                                                 */
/*   Requirements : If invoked in the TSO/E address space,         */
/*                  SYSEXEC DD must be assigned to dataset         */
/*                  in which this EXEC resides.                    */
/*                                                                 */
/*                                                                 */
/*   Changed :                                                     */
/*******************************************************************/

/*  Environment                                                    */
 NUMERIC DIGITS 2Ø;

/*  Constants:                                                     */
 inpdd='BSDSIN';            /* DDNAME from the JOB                 */
 prf='DSNP31Ø.LOGCOPYX';    /* Prefix for DB2 Active Log VSAMs     */
 bl=' ';                    /* One space String                    */
 zr='Ø';
 dt='.';
 cm='.';

 SAY '*********************************************************';
 SAY '* EXEC for displaying Active Log status on operators console*';
 SAY '*********************************************************';
 SAY '';
 SAY '* Processing started...                                 *';
 SAY '* Date: '||date()||'     Time: '||time();
 SAY '';

 SAY '*********************************************************';
 SAY '* Input string processing - Fetching DB2 SSID           *';
 SAY '*********************************************************';
 SAY '';
```

```
PARSE UPPER EXTERNAL ssid;
IF ssid='' THEN DO;
   ssid='DSN';
   SAY 'DB2 SSID not specified. Default is DSN.';
END;

SAY '**************************************************************';
SAY '* Reading DSNJUØØ4 output                                    *';
SAY '**************************************************************';
SAY '';

DROP inl.;
"EXECIO * DISKR "inpdd" (STEM inl. FINIS)"
IF rc ¨=Ø THEN EXIT rc;
IF inl.Ø=Ø THEN DO;
   SAY 'Input file 'inpdd' is empty ';
   EXIT 16;
END;

SAY '**************************************************************';
SAY '* Processing DSNJUØØ4 output                                 *';
SAY '**************************************************************';
SAY '';

v.='Unknown';
DROP outl.;
j=Ø;i=Ø;
logcopy=Ø;
j=j+1;outl.j='DISPLAY DB2 ACTIVE LOG STATUS. DB2 SSID: 'ssid;
DO WHILE i<=inl.Ø;
   i=i+1;
   inl.i=substr(inl.i,2);
   SELECT;
   WHEN subword(inl.i,1,3)='LOG MAP OF' THEN DO;
      j=j+1;outl.j=bl||strip(inl.i);
   END;
   WHEN subword(inl.i,1,3)='HIGHEST RBA WRITTEN' THEN DO;
      v.hrbawrt=word(inl.i,4);
      lng=wordindex(inl.i,5)-1;
      v.hrbawdt=trandate(word(inl.i,5));
      v.hrbawtm=word(inl.i,6);
   END;
   WHEN subword(inl.i,1,3)='HIGHEST RBA OFFLOADED' THEN DO;
      v.hrbaoff=word(inl.i,4);
   END;
   WHEN subword(inl.i,1,3)='ACTIVE LOG COPY' THEN DO;
      j=j+1;outl.j=strip(inl.i,'T');
      logcopy=word(inl.i,4);
      i=i+2;
      logno=Ø;
      j=j+1;outl.j=bl;
      outl.j=insert('DS#',outl.j,1);
      outl.j=insert('FULL%',outl.j,9);
      outl.j=insert('STATUS',outl.j,19);
   END;
   WHEN datatype(word(inl.i,1),X) & logcopy¨=Ø THEN DO;
      logno=logno+1;
      v.rba1.logcopy.logno=word(inl.i,1);
      v.rba2.logcopy.logno=word(inl.i,2);
      v.crdt.logcopy.logno=trandate(word(inl.i,3));
      v.crtm.logcopy.logno=word(inl.i,4);
      v.dsnm.logcopy.logno=substr(word(inl.i,5),5);
      v.dsno.logcopy.logno=substr(word(inl.i,5),length(prf)+6);
      i=i+1;
      v.dat1.logcopy.logno=trandate(word(inl.i,1));
      v.tim1.logcopy.logno=word(inl.i,2);
      v.dat2.logcopy.logno=trandate(word(inl.i,3));
      v.tim2.logcopy.logno=word(inl.i,4);
      v.pass.logcopy.logno=word(inl.i,5);
      v.stat.logcopy.logno=substr(word(inl.i,6),8);
      tmp=translate(v.stat.logcopy.logno,bl,cm);
      v.sta1.logcopy.logno=word(tmp,1);
      IF words(tmp)>1 THEN v.sta2.logcopy.logno=word(tmp,2);
      ELSE v.sta2.logcopy.logno=bl;
      SELECT;
      WHEN v.stat.logcopy.logno='NEW' |,
           v.stat.logcopy.logno='REUSABLE' THEN,
           v.pctf.logcopy.logno='  Ø.ØØ';
      WHEN index(v.stat.logcopy.logno,'STOPPED')¨=Ø |,
           index(v.stat.logcopy.logno,'TRUNCATED')¨=Ø THEN,
           v.pctf.logcopy.logno='1ØØ.ØØ';
      OTHERWISE DO;
           IF x2b(v.rba1.logcopy.logno) <= x2b(v.hrbawrt) &,
              x2b(v.rba2.logcopy.logno) >= x2b(v.hrbawrt) THEN DO;
              v.pctf.logcopy.logno=format(,
                (x2d(v.hrbawrt)-x2d(v.rba1.logcopy.logno)) /,
                (x2d(v.rba2.logcopy.logno)-x2d(v.rba1.logcopy.logno))*,
                1ØØ,3,2);
           END;
           ELSE v.pctf.logcopy.logno='1ØØ.ØØ';
      END;
      END;
      v.pctf.logcopy.logno=v.pctf.logcopy.logno||'%';
      j=j+1; outl.j=bl;
      outl.j=insert(v.dsno.logcopy.logno,outl.j,1);
      outl.j=insert(v.pctf.logcopy.logno,outl.j,7);
      outl.j=insert(v.stat.logcopy.logno,outl.j,19);
   END;
   OTHERWISE logcopy=Ø;
   END;
END;

SAY '**************************************************************';
SAY '* Sending created output                                     *';
SAY '**************************************************************';
SAY '';

DO i=1 to j;
   ADDRESS TSO "SEND '"outl.i"'";
END;

EXIT Ø;

trandate: PROCEDURE EXPOSE zr bl dt;
   PARSE UPPER ARG indate;
   IF verify(indate,dt)=Ø THEN RETURN copies(dt,1Ø);
   gg=substr(indate,1,2);
   ddd=substr(indate,4,3);
   IF gg<'7Ø' THEN gggg='2Ø'gg;
   ELSE            gggg='19'gg;
   g=gggg;d=ddd;
   m.Ø=12;m.1=31;m.2=28;m.3=31;m.4=3Ø;m.5=31;m.6=3Ø;
   m.7=31;m.8=31;m.9=3Ø;m.1Ø=31;m.11=3Ø;m.12=31;
   IF mod(g,4)=Ø & mod(g,1ØØ)¬=Ø | mod(g,4ØØ)=Ø THEN m.2=29;
   dd=Ø;
   mm=Ø;
   DO WHILE dd=Ø;
      mm=mm+1;
      IF d-m.mm<=Ø THEN dd=d;
      ELSE d=d-m.mm;
   END;
   tmp=gggg||'-'||format(mm,2)||'-'||format(dd,2);
   outdate=translate(tmp,zr,bl);
RETURN outdate;

mod: PROCEDURE;
   PARSE UPPER ARG god,div;
   dp=god/div-trunc(god/div);
RETURN dp;
```

ENVIRONMENT

This procedure runs under MVS/ESA, JES3, and DB2 Version 3.

AUTHORIZATION

No DB2 authorization is required to execute this job because the job can be executed even when DB2 is not running. You must check with the JES, MVS, and RACF administrator for the following:

- The user-id of the user who calls the job from the JES console from the disk reader, and the JES command used for calling the job.
- The user-id that has the read access to the BSDSs and other datasets noted in the JOB.
- The user-id that has privileges for executing the TSO SEND command.

Josip Ivancic
Database Administrator
SDA (Croatia)

SMART REORG

GENERAL INFORMATION

This tool helps in identifying and REORGing the appropriate partition of a partitioned tablespace. This is best illustrated in a scenario where a weekly or monthly job is scheduled which populates various business entities. The table under consideration is partitioned and these records will be loaded into the particular partition.

BENEFITS

The REORG helps in the clustering of the data and the indexes according to the definition. The updated catalog statistics help the Optimizer to choose the correct access path based on the latest status.

The SMART REORG contains two components. These are:

- PARTITION

- REORG.

The partition member in VCAT.DBAXX.REXX is executed first and will in turn execute the REORG REXX.

PARTITION

```
/* REXX*/
/*******************************************************************/
/*                                                                 */
/* THIS REXX WILL GET THE PARTITION VALUE BASED ON THE DATE GIVEN     */
/* IN THE INPUT FILE. THE PARTITION VALUE IS OBTAINED BY A QUERY ON   */
/* SYSTABLEPART  WITH THE DATE VALUE IN THE LIMIT KEY.             */
/*                                                                 */
/* THE DATE IS TAKEN AS THE INPUT BECAUSE THE TABLESPACE IS         */
/* PARTITIONED ON YEAR AND MONTH                                   */
/*                                                                 */
/* THE SECOND STEP IS TO GET THE DEADLINE TIMESTAMP FROM A VIEW       */
/* ON SYSDUMMY TABLE                                               */
/*                                                                 */
/* WITH THE ABOVE TWO INPUTS THE THIRD STEP EXECUTES THE REORG       */
/*                                                                 */
/*                                                                 */
/* ASSUMPTIONS  :1 THE DATABASE AND THE TABLESPACE NAMES ARE GIVEN BY */
/*                THE USER AND ARE EIGHT CHARACTERS IN LENGTH       */
/*                ie DBXXXXXX and TSXXXXXX                         */
/*                                                                 */
/*                                                                 */
/*              2 SYSDUMMY FOR GETTING CURRENT TIMESTAMP. IT HAS A   */
/*                VIEW DEFINED ON IT.                              */
/*                IT IS (CURRENT TIMESTAMP + 4 HOURS)              */
/*                A SELECT IS MADE TO THE VIEW SYSDUMMY AS IT       */
/*                RETURNS ONE ROW ONLY.                            */
/*                                                                 */
/*                                                                 */
/*              3 GDG BASE SHOULD EXIST FOR THE IMAGECOPY STEP AND   */
/*                THE SAME GDG IS USED FOR INLINE IMAGECOPY DURING   */
/*                REORG                                            */
/*******************************************************************/

/* READ INPUT FILE INTO STEM                                      */

INDD1="'VCAT.DBA.UNLD'"
"ALLOC DD(IN1) DSN("INDD1") SHR REUSE"
"EXECIO * DISKR IN1  (STEM INLIST. FINIS"
"FREE DD(IN1)"

/* GET THE DATE IN THE FORM CCYYMM                                */
/* THE REASON FOR GETTING THE DATE IN CCYYMM FORM IS THE LIMIT KEY  */
/* IN SYSTABLEPART IS ASSUMED TO BE OF CCYYMM FORMAT                */
/* THE SUBSTRING IS DONE TO GET THE CORRECT MONTH POSITION          */
/* EXAMPLE - THE DATE IN INPUT FILE WILL BE OF THE FORM -:          */
/*                                                                  */
/*            CCYY   MM                                             */
/*           Ø2ØØ5ØØØØ9ØØØ45                                        */

IF SUBSTR(INLIST.1,9,1) == Ø THEN DO

/* CASE 1 : WHEN THE MONTH FALLS IN THE RANGE Ø1 TO Ø9              */

   VAL = (SUBSTR(INLIST.1,2,4)]]','||SUBSTR(INLIST.1,1Ø,1))
END

ELSE DO

/* CASE 2 : WHEN THE MONTH FALLS IN THE RANGE 1Ø TO 12              */

  VAL = (SUBSTR(INLIST.1,2,4)]]','||SUBSTR(INLIST.1,9,2))
END

/* DISPLAY THE DATE                                                 */

SAY 'VALUE : ' VAL
SAY 'HERE GOES THE SQL!!!!'

/* QUEUE THE JCL TO A SEQUENTIAL DATASET FOR SUBMISSION             */

"ALLOC DS(DBA.REPORT)  FI(OUT) SHR"
NEWSTACK
QUEUE "//PSYSDBAS  JOB (SYSØØØØØØØ,B4B),DBAS-DBAXX-SHOPS,CLASS=6, "
QUEUE "//    MSGCLASS=X,NOTIFY=&SYSUID,REGION=2Ø48K,MSGLEVEL=(1,1)  "
QUEUE "//*********************************************************"
QUEUE "//*                                                        "
QUEUE "//* JCL TO DO BACKGROUND DYNAMIC DB2 EXECUTION             "
QUEUE "//*********************************************************"
QUEUE "//STEPØ1Ø EXEC PGM=IKJEFTØ1,DYNAMNBR=2Ø,COND=(4,LT)        "
QUEUE "//STEPLIB DD DSN=SYS2.DB2.DBAP.DSNLOAD,DISP=SHR            "
QUEUE "//SYSTSPRT DD SYSOUT=*                                     "
QUEUE "//SYSPRINT DD SYSOUT=*                                     "
QUEUE "//SYSOUT   DD SYSOUT=*                                     "
QUEUE "//SYSRECØØ DD DSN=VCAT.DBA.DATA,DISP=SHR                   "
QUEUE "//SYSPUNCH DD DUMMY                                        "
QUEUE "//SYSTSIN  DD *                                            "
QUEUE "    DSN SYSTEM(DBAP)                                       "
QUEUE "   RUN PROGRAM(DSNTIAUL) PARMS('SQL')                      "
QUEUE "   END                                                     "
QUEUE "/*                                                         "
QUEUE "//SYSIN    DD *                                            "
QUEUE "   SET CURRENT DEGREE = 'ANY' ;                                 "
QUEUE "   SELECT DIGITS(PARTITION) FROM SYSIBM.SYSTABLEPART            "
QUEUE "   WHERE TSNAME = 'TSXXXXXX'                                    "
QUEUE "   AND LIMITKEY LIKE  '"VAL"%';                                 "
QUEUE "/*                                                              "
QUEUE "//*                                                             "
QUEUE "//STEPØ2Ø EXEC PGM=IKJEFTØ1,DYNAMNBR=2Ø,COND=(4,LT)             "
QUEUE "//STEPLIB DD DSN=SYS2.DB2.DBAP.DSNLOAD,DISP=SHR                 "
QUEUE "//SYSTSPRT DD SYSOUT=*                                          "
QUEUE "//SYSPRINT DD SYSOUT=*                                          "
QUEUE "//SYSOUT   DD SYSOUT=*                                          "
QUEUE "//SYSRECØØ DD DSN=VCAT.DBA.DATE,DISP=SHR                        "
QUEUE "//SYSPUNCH DD DUMMY                                             "
QUEUE "//SYSTSIN  DD *                                                 "
QUEUE "    DSN SYSTEM(DBAP)                                            "
QUEUE "   RUN PROGRAM(DSNTIAUL) PARMS('SQL')                           "
QUEUE "   END                                                          "
QUEUE "/*                                                              "
QUEUE "//SYSIN    DD *                                                 "
QUEUE "   SET CURRENT DEGREE = 'ANY' ;                                 "
QUEUE "   SELECT DB2_TS  FROM DB2.SYSDUMMY_VIEW;                       "
QUEUE "/*                                                              "
QUEUE "//*                                                             "
QUEUE "//STEP3Ø   EXEC PGM=IKJEFTØ1,REGION=4Ø96K                       "
QUEUE "//SYSEXEC  DD DSN=VCAT.DBAXX.REXX,DISP=SHR                      "
QUEUE "//SYSTSPRT DD SYSOUT=*                                          "
QUEUE "//SYSTSIN DD *                                                  "
QUEUE "   EXECUTIL SEARCHDD(YES)                                       "
QUEUE "   %REORG                                                       "
QUEUE "//*                                                             "
"EXECIO 49 DISKW OUT (FINIS"
"SUBMIT 'VCAT.DBA.REPORT'"
"FREE DD(OUT)"
DELSTACK

/*                                                                   */
/*******************************************************************/
/* OUTPUT                                                            */
/*                                                                   */
/*    1 PART - : "ØØØPART"   LIKE ØØØ35 OR ØØ134                      */
/*                                                                   */
/*    2 TIMESTAMP - :  1998-11-23-Ø6.13.32.924895                    */
/*                                                                   */
/*******************************************************************/
```

REORG

```
/* REXX*/
/********************************************************************/
```

```
/* THIS REXX DOES THE REORG GETTING THE PART AND DEADLINE TIMESTAMP  */
/* FROM THE INPUT OF THE PREVIOUS JOBS                               */
/*                                                                   */
/* ASSUMPTIONS : THE DATABASE AND SPACENAME ARE GIVEN BY THE USER    */
/*               THE NUMCOLS AND COUNT IN REORG ARE ASSUMED TO BE    */
/*               OF VALUE 3 AND 1Ø WHICH CAN BE CHANGED BY THE USER  */
/*               DEPENDING ON THE REQUIREMENT                        */
/*********************************************************************/
/* REASONS : -THE REASON FOR USING SORTDATA, NOSYSREC, AND SORTKEYS  */
/*            IN THE REORG IS BECAUSE THE TABLESPACE IS PARTITIONED, */
/*            THE DATASET IS IN CLUSTERING ORDER, THE OUTPUT OF      */
/*            SORT IS THE INPUT FOR RELOAD AND THE INDEX KEYS WILL   */
/*            BE SORTED IN PARALLEL WITH THE RELOAD                  */
/*********************************************************************/

/* READ INPUT FOR PART VALUE                                         */

INDD1="'VCAT.DBA.DATA'"
"ALLOC DD(IN1) DSN("INDD1") SHR REUSE"
"EXECIO * DISKR IN1  (STEM INPUT. FINIS"
"FREE DD(IN1)"

/* THE REASON FOR DOING THE SUBSTRING IS THE PART VALUE FROM         */
/* THE SYSTABLEPART IS OF THE FORM "ØØØPART" LIKE "ØØØ32" OR         */
/* LIKE "ØØ132"                                                      */

IF SUBSTR(INPUT.1,3,1) == Ø THEN DO /* CASE 1: PART VALUE LIKE ØØØ32 */
   PART = SUBSTR(INPUT.1,4,2)
END
ELSE DO
   PART = SUBSTR(INPUT.1,3,3)       /* CASE 2: PART VALUE LIKE ØØ132 */
END

/* READ INPUT FOR DEADLINE                                          */

INDD2="'VCAT.DBA.DATE'"
"ALLOC DD(IN2) DSN("INDD2") SHR REUSE"
"EXECIO * DISKR IN2  (STEM INPUTS. FINIS"
"FREE DD(IN2)"
DATA = SUBSTR(INPUTS.1,1,26)
SAY 'DATA : ' DATA
SAY 'HERE GOES THE REORG!!!!'

/* QUEUE THE JCL FOR REORG                                          */

"ALLOC DS(DBA.REORG.JCL)  FI(OUT) SHR"
NEWSTACK
QUEUE "//PSYSDBAS  JOB (SYSØØØØØØØ,B4B),DBAS-DBAXX-SYSTEMS,CLASS=6, "
QUEUE "//  MSGCLASS=X,REGION=4Ø96K,MSGLEVEL=(1,1),NOTIFY=&SYSUID    "
QUEUE "/*JOBPARM LINECT=60                                              "
QUEUE "//*                                                              "
QUEUE "//* ****************************************************** "
QUEUE "//* *  JCL FOR REORG UTILITY                                   * "
QUEUE "//* ****************************************************** "
QUEUE "//STARTB   EXEC PGM=IKJEFT01,DYNAMNBR=20                         "
QUEUE "//SYSTSPRT DD SYSOUT=*                                           "
QUEUE "//SYSPRINT DD SYSOUT=*                                           "
QUEUE "//SYSTSIN  DD *                                                  "
QUEUE "  DSN SYSTEM(DBAP)                                               "
QUEUE "  -START DATABASE(DBXXXXXX) SPACENAM(TSXXXXXX) -                 "
QUEUE "   PART("PART") ACCESS(UT)                                       "
QUEUE "//*                                                              "
QUEUE "//IMAGCPY  EXEC PGM=DSNUTILB,TIME=1439,                          "
QUEUE "//         PARM='DBAP,REORPPXX'                                  "
QUEUE "//SYSCOPY  DD UNIT=CART,DISP=(,CATLG),                           "
QUEUE "//         DCB=(DSCB1,RECFM=FB,LRECL=4096,BLKSIZE=28672),        "
QUEUE "//         VOL=(,,,99),                                          "
QUEUE "//         DSNAME=VCAT.DBXXXXXX.TSXXXXXX.F(+1),                  "
QUEUE "//         LABEL=(1,SL,EXPDT=99000)                              "
QUEUE "//SYSPRINT DD SYSOUT=*                                           "
QUEUE "//SYSUDUMP DD SYSOUT=*                                           "
QUEUE "//SYSIN    DD *                                                  "
QUEUE "  COPY TABLESPACE DBXXXXXX.TSXXXXXX  COPYDDN SYSCOPY             "
QUEUE "    FULL YES SHRLEVEL REFERENCE                                  "
QUEUE "//*                                                              "
QUEUE "//UTIL     EXEC PROC=DSNUPROC,TIME=1439,                         "
QUEUE "//         SYSTEM=DBAP,UID='REORPPXX',UTPROC=''                  "
QUEUE "//DSNUPROC.SYSPRINT DD SYSOUT=*                                  "
QUEUE "//DSNUPROC.UTPRINT  DD SYSOUT=*                                  "
QUEUE "//DSNUPROC.SYSUDUMP DD DUMMY                                     "
QUEUE "//DSNUPROC.SORTWK01 DD DSNAME=VCAT.DBXXXXXX.TSXXXXXX.SORTW1, "
QUEUE "//         DISP=(MOD,DELETE),SPACE=(CYL,(100,30))                "
QUEUE "//*                                                              "
QUEUE "//DSNUPROC.SORTWK02 DD DSNAME=VCAT.DBXXXXXX.TSXXXXXX.SORTW2, "
QUEUE "//         DISP=(MOD,DELETE),SPACE=(CYL,(100,30))                "
QUEUE "//*                                                              "
QUEUE "//DSNUPROC.SORTWK03 DD DSNAME=VCAT.DBXXXXXX.TSXXXXXX.SORTW3, "
QUEUE "//         DISP=(MOD,DELETE),SPACE=(CYL,(100,30))                "
QUEUE "//*                                                              "
QUEUE "//DSNUPROC.SORTWK04 DD DSNAME=VCAT.DBXXXXXX.TSXXXXXX.SORTW4, "
QUEUE "//         DISP=(MOD,DELETE),SPACE=(CYL,(100,30))                "
QUEUE "//*                                                              "
QUEUE "//SYSCOPY1 DD UNIT=CART,DISP=(,CATLG),                           "
QUEUE "//         DCB=(DSCB1,RECFM=FB,LRECL=4096,BLKSIZE=28672),        "
QUEUE "//         VOL=(,,,99),                                          "
QUEUE "//         DSNAME=VCAT.DBXXXXXX.TSXXXXXX.F(+2),                  "
QUEUE "//         LABEL=(1,SL,EXPDT=99000)                              "
QUEUE "//*SYSREC   DD DSNAME=VCAT.DBXXXXXX.TSXXXXXX.UNLD,               "
QUEUE "//*         DISP=(,DELETE,CATLG),SPACE=(CYL,(2ØØ,3Ø),RLSE)   "
QUEUE "//*                                                          "
QUEUE "//*SYSUT1   DD DSNAME=VCAT.DBXXXXXX.TSXXXXXX.SYSUT1,         "
QUEUE "//*         DISP=(,DELETE,CATLG),SPACE=(CYL,(1ØØ,3Ø),RLSE)   "
QUEUE "//*                                                          "
QUEUE "//*SORTOUT  DD DSNAME=VCAT.DBXXXXXX.TSXXXXXX.SORTOUT,        "
QUEUE "//*         DISP=(,DELETE,CATLG),SPACE=(CYL,(1ØØ,3Ø),RLSE)   "
QUEUE "//********************************************************* "
QUEUE "//*THE FOLLOWING CONTROL CARD IS TABLESPACE YOU ARE REORGING "
QUEUE "//**********************************************************"
QUEUE "//SYSIN    DD *                                              "
QUEUE "  REORG TABLESPACE DBXXXXXX.DBXXXXXX PART "PART"             "
QUEUE "  LOG NO SORTDATA                                            "
QUEUE "  SORTKEYS NOSYSREC  COPYDDN(SYSCOPY1) SHRLEVEL REFERENCE    "
QUEUE "  DEADLINE "DATA"                                            "
QUEUE "/*                                                           "
QUEUE "//STARTA   EXEC PGM=IKJEFTØ1,DYNAMNBR=2Ø                     "
QUEUE "//SYSTSPRT DD SYSOUT=*                                       "
QUEUE "//SYSPRINT DD SYSOUT=*                                       "
QUEUE "//SYSTSIN  DD *                                              "
QUEUE "  DSN SYSTEM(DBAP)                                           "
QUEUE "  -START DATABASE(DBXXXXXX) SPACENAM(TSXXXXXX) -             "
QUEUE "  PART("PART")  ACCESS(RW)                                   "
QUEUE "/*                                                           "
QUEUE "//*                                                          "
QUEUE "//STEP1Ø   EXEC PGM=DSNUTILB,TIME=1439,                      "
QUEUE "//         PARM='DBAP,RUNSTATS'                              "
QUEUE "//* ********************************************************"
QUEUE "//* **  PARM HAS THE FORMAT DBAT  = DB2 SUBSYSTEM ID         "
QUEUE "//* **         RUNSTATS = DB2 UTILITY ID                     "
QUEUE "//* ********************************************************"
QUEUE "//SYSPRINT DD SYSOUT=*                                       "
QUEUE "//SYSIN    DD *                                              "
QUEUE "   RUNSTATS TABLESPACE DBXXXXXX.TSXXXXXX PART "PART"         "
QUEUE "   TABLE(ALL) INDEX(ALL) KEYCARD FREQVAL NUMCOLS(3) COUNT(1Ø)"
QUEUE "   SHRLEVEL REFERENCE UPDATE(ALL)                            "
QUEUE "/*                                                           "
"EXECIO 93 DISKW OUT (FINIS"
"FREE DD(OUT)"
"SUBMIT 'VCAT.DBA.REORG.JCL'"
DELSTACK
EXIT
```

Kiran Haryadi and K R Swaminaathan
DBA
Wipro Infotech (India)

DB2 utility services – part 4

This month we conclude the article on DB2 services, which provide an easy way to execute any DB2 utility.

DBREORG

```
)TBA 72)CM ————————————————————————————————————————————————————————————
)CM Skeleton to generate JCL for DB2 utility                          –
)CM ———————————————————————————————————————————————————————————————————
//&user.X JOB (ACCT#),'&option',
//             NOTIFY=&user,REGION=4M,
//             CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//* *******************************************************************
//*    &title
//*    GENERATION DATE AND TIME : &date AT: &time
//*
//*    CALCULATING TIME IS &ctime SECONDS.
//*
//*    REORG - WAS RUN WITH THE FOLLOWING PARAMETERS:
//*    PARAMETER   PARAMETER VALUE
//*    —————   ———————————
//*    SSID      :  &db2
//*    Creator   :  &creC
//*    Name      :  &tabc
//*    Tsname    :  &tsnc
//*    Dbname    :  &dbnc
//*    Icopy     :  &ico
//*    Copypref  :  &pref
//*    Log       :  &log
//*    Sortdata  :  &sor
//*    Keepdict  :  &dic
//*    Stopts    :  &sto
//*    Quiesce   :  &que
//*    Runstat   :  &rru
//*    Volume    :  &vol  &vol1  &vol2
//*    Catname   :  &catn
//*    Tracks    :  &trk
//* *******************************************************************
//*  NUM  DATABASE  TABLESPACE   TRACKS   PART
//*  –    ————      —————        ———      ——
)DOT "ALIST"
//* &detail
)ENDDOT
//*  –    ————      —————        ———      ——
//*                        TOTAL:&tot TRACKS OR
//*                             &cyl CYLINDERS
//*—————————————————————————
//*
)SEL &sto = YES
//*—— STOP TABLESPACES ———————————————
//STOPTS   EXEC PGM=IKJEFTØ1,COND=(4,LT)
//STEPLIB  DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
)BLANK 1
)DOT "ALIST"
)SEL &pts EQ S
   -STOP  DATABASE(&db) SPACENAM(&ts)
   -START DATABASE(&db) SPACENAM(&ts) ACCESS(UT)
)ENDSEL
)SEL &pts NE S
   -STOP  DATABASE(&db) SPACENAM(&ts) PART(&pts)
   -START DATABASE(&db) SPACENAM(&ts) PART(&pts) ACCESS(UT)
)ENDSEL
)BLANK 1
)ENDDOT
//*
)ENDSEL
)SEL &que = YES
//*—— TERMINATE UTILITY ———————————————
//TERMQB   EXEC PGM=IKJEFTØ1,COND=(4,LT)
//STEPLIB  DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
   -TERM UTILITY(&user..QUIEUB)
/*
//*—————————————————————————
//*—— QUIESCE STEP                          —————
//QUIEUB   EXEC DSNUPROC,SYSTEM=&db2,
//      UID='&user..QUIEUB',UTPROC='',COND=(4,LT)
//STEPLIB  DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
//SYSIN    DD  *
  QUIESCE
)DOT "ALIST"
)SEL &pts EQ S
     TABLESPACE &DB..&TS
)ENDSEL
)SEL &pts NE S
     TABLESPACE &DB..&TS PART &pts
)ENDSEL
)ENDDOT
/*
)ENDSEL
```

```
)SEL &ico = BEFORE | &ico = BOTH
//*—— TERMINATE UTILITY - BEFORE IMAGE COPY ————
//TERMCB   EXEC PGM=IKJEFTØ1,COND=(4,LT)
//STEPLIB  DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
   -TERM UTILITY(&user..COPYB)
   END
/*
//*—— BEFORE IMAGE COPY  ——————————————
//COPYB    EXEC DSNUPROC,SYSTEM=&db2,REGION=4Ø96K,
//         UID='&user..COPYB',UTPROC='',COND=(4,LT)
//STEPLIB DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
)DOT "ALIST"
//COPB&scu  DD UNIT=339Ø,VOL=SER=&vol,
)SEL &pts EQ S
//        DSN=&pref..&db..&ts..&dsufb,
)ENDSEL
)SEL &pts NE S
//        DSN=&pref..&db..&ts..P&pts..&dsufb,
)ENDSEL
//        DCB=(BUFNO=2Ø,BLKSIZE=22528),
//        DISP=(,CATLG,DELETE),
//        SPACE=(TRK,(&pri,&sec,),RLSE)
)ENDDOT
//SYSIN  DD  *
)DOT "ALIST"
)BLANK 1
  COPY  TABLESPACE &db..&ts
)SEL &pts NE S
             DSNUM &pts
)ENDSEL
           COPYDDN COPB&scu
              FULL YES
          SHRLEVEL REFERENCE
)ENDDOT
/*
)ENDSEL
)DOT "ALIST"
//*—— TERMINATE UTILITY - BEFORE REORG      ————
//TROR&scu   EXEC PGM=IKJEFTØ1,COND=(4,LT)
//STEPLIB  DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
   -TERM UTILITY(&user..REOR&scu)
   END
/*
)SEL &pts EQ S
```

```
//*—— REORG &db..&ts
)ENDSEL
)SEL &pts NE S
//*—— REORG &db..&ts PART &pts
)ENDSEL
//REOR&scu  EXEC DSNUPROC,SYSTEM=&db2,REGION=4Ø96K,
//      UID='&user..REOR&scu',UTPROC=''
//STEPLIB DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
//SYSREC  DD UNIT=339Ø,VOL=SER=&vol,
)SEL &pts EQ S
//    DSN=&user..&db..&ts..REC.&dsufb,
)ENDSEL
)SEL &pts NE S
//    DSN=&user..&db..&ts..P&pts..REC.&dsufb,
)ENDSEL
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,CATLG)
//SORTOUT DD UNIT=339Ø,VOL=SER=&vol,
)SEL &pts EQ S
//    DSN=&user..&db..&ts..OUT.&dsufb,
)ENDSEL
)SEL &pts NE S
//    DSN=&user..&db..&ts..P&pts..OUT.&dsufb,
)ENDSEL
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,CATLG)
//SYSUT1  DD UNIT=339Ø,VOL=SER=&vol,
)SEL &pts EQ S
//    DSN=&user..&db..&ts..UT1.&dsufb,
)ENDSEL
)SEL &pts NE S
//    DSN=&user..&db..&ts..P&pts..UT1.&dsufb,
)ENDSEL
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,CATLG)
//SORTWKØ1 DD UNIT=339Ø,VOL=SER=&vola,
)SEL &pts EQ S
//    DSN=&user..&db..&ts..WK1.&dsufb,
)ENDSEL
)SEL &pts NE S
//    DSN=&user..&db..&ts..P&pts..WK1.&dsufb,
)ENDSEL
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,DELETE),
//    DCB=(BUFNO=1Ø)
//SORTWKØ2 DD UNIT=339Ø,VOL=SER=&volb,
)SEL &pts EQ S
//    DSN=&user..&db..&ts..WK2.&dsufb,
)ENDSEL
)SEL &pts NE S
```

```
//    DSN=&user..&db..&ts..P&pts..WK2.&dsufb,
)ENDSEL
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,DELETE),
//    DCB=(BUFNO=1Ø)
//SORTWKØ3 DD UNIT=339Ø,VOL=SER=&vola,
)SEL &pts EQ S
//    DSN=&user..&db..&ts..WK3.&dsufb,
)ENDSEL
)SEL &pts NE S
//    DSN=&user..&db..&ts..P&pts..WK3.&dsufb,
)ENDSEL
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,DELETE),
//    DCB=(BUFNO=1Ø)
//SORTWKØ4 DD UNIT=339Ø,VOL=SER=&volb,
)SEL &pts EQ S
//    DSN=&user..&db..&ts..WK4.&dsufb,
)ENDSEL
)SEL &pts NE S
//    DSN=&user..&db..&ts..P&pts..WK4.&dsufb,
)ENDSEL
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,DELETE),
//    DCB=(BUFNO=1Ø)
//SYSIN    DD  *
)BLANK 1
  REORG    TABLESPACE &db..&ts
                  LOG &log
)SEL &pts NE S
                 PART &pts
)ENDSEL
)SEL &sor = YES
             SORTDATA
)ENDSEL
)SEL &dic = YES
       KEEPDICTIONARY
)ENDSEL
)SEL &rru = YES
)BLANK 1
  RUNSTATS TABLESPACE &db..&ts
)SEL &pts NE S
                 PART &pts
)ENDSEL
                INDEX (ALL)
             SHRLEVEL REFERENCE
)BLANK 1
)ENDSEL
)ENDDOT
)SEL &ico = AFTER | &ico = BOTH
```

```
//*—— TERMINATE UTILITY - AFTER IMAGE COPY ————
//TERMCA EXEC PGM=IKJEFTØ1,COND=(4,LT)
//STEPLIB  DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
   -TERM UTILITY(&user..COPYA)
   END
/*
//*—— AFTER  IMAGE COPY  ——————————
//COPYA EXEC DSNUPROC,SYSTEM=&db2,REGION=4Ø96K,
//      UID='&user..COPYA',UTPROC='',COND=(4,LT)
//STEPLIB DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
)DOT "ALIST"
//COPA&scu DD UNIT=339Ø,VOL=SER=&vol,
)SEL &pts EQ S
//    DSN=&pref..&db..&ts..&dsufa,
)ENDSEL
)SEL &pts NE S
//    DSN=&pref..&db..&ts..P&pts..&dsufa,
)ENDSEL
//    DCB=(BUFNO=2Ø,BLKSIZE=22528),
//    DISP=(,CATLG,DELETE),
//    SPACE=(TRK,(&pri,&sec,),RLSE)
)ENDDOT
//SYSIN  DD  *
)DOT "ALIST"
)BLANK 1
  COPY  TABLESPACE &db..&ts
)SEL &pts NE S
             DSNUM &pts
)ENDSEL
           COPYDDN COPA&scu
              FULL YES
          SHRLEVEL REFERENCE
)ENDDOT
)ENDSEL
/*
)SEL &que = YES
//*—— TERMINATE UTILITY ——————————
//TERMQA  EXEC PGM=IKJEFTØ1,COND=(4,LT)
//STEPLIB  DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
   -TERM UTILITY(&user..QUIEUA)
/*
//*————————————————————————————
//*—— QUIESCE STEP                         ————
//QUIEUA EXEC DSNUPROC,SYSTEM=&db2,
//      UID='&user..QUIEUA',UTPROC='',COND=(4,LT)
//STEPLIB  DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
//SYSIN    DD  *
  QUIESCE
)DOT "ALIST"
)SEL &pts EQ S
     TABLESPACE &DB..&TS
)ENDSEL
)SEL &pts NE S
     TABLESPACE &DB..&TS PART &pts
)ENDSEL
)ENDDOT
/*
)ENDSEL
)SEL &sto = YES
//*—— START TABLESPACES ——————————————
//STARTS  EXEC PGM=IKJEFTØ1,COND=(4,LT)
//STEPLIB  DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
)BLANK 1
)DOT "ALIST"
)SEL &pts EQ S
   -START DATABASE(&db) SPACENAM(&ts) ACCESS(RW)
)ENDSEL
)SEL &pts NE S
   -START DATABASE(&db) SPACENAM(&ts) PART(&pts) ACCESS(RW)
)ENDSEL
)ENDDOT
//*
)ENDSEL
```

DBSTOS

```
)TBA 72)CM ————————————————————————————————————————————————————————————
)CM Skeleton to generate JCL for DB2 utility                          —
)CM ———————————————————————————————————————————————————————————————————
//&user.X JOB (ACCT#),'&option',
//             NOTIFY=&user,REGION=4M,
//             CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//* ******************************************************************
//*    &title
//*    GENERATION DATE AND TIME : &date AT: &time
//*
//*    STOSPACE - WAS RUN WITH THE FOLLOWING PARAMETERS:
//*    PARAMETER   PARAMETER VALUE
//*    —————  ———————————
//*    SSID     :  &db2
//*    All      :  &sga
//*    Creator  :  &scre
//*    Sgname   :  &sgn
//* *******************************************************************
//*
//*—— STOSPACE ——————————————————
//STOSPA EXEC DSNUPROC,SYSTEM=&db2,
//      UID='&user..STOSPA',UTPROC=''
//STEPLIB  DD DSN=DSN410.SDSNLOAD,DISP=SHR
//SYSIN    DD  *
)SEL &sga = YES
  STOSPACE STOGROUP(*)
)ENDSEL
)SEL &sga = NO
)DOT "SLIST"
  STOSPACE STOGROUP &sgname
)ENDDOT
)ENDSEL
/*
```

PDBUTIL

```
* PROCESS GS,OFFSET,OPT(TIME);PDBCOPY:PROC(PARMS)OPTIONS(MAIN) REORDER;
/****************************************************************/
/* DESCRIPTION: PL/I PROGRAM FOR UTILITY                        */
/****************************************************************/
  DCL PARMS CHAR(100) VAR;
  DCL SYSPRINT    FILE STREAM OUTPUT;
  DCL NUMSEQ        BIN FIXED(31) INIT(0);
  DCL HCARD         BIN FIXED(31);
  DCL CARD          BIN FIXED(31);
  DCL MCARD         PIC'-.-.-.-9';
  DCL HTBNAME       CHAR(18);
  DCL HTBCREATOR    CHAR(8);
  DCL HDBNAME       CHAR(8)  VAR;
  DCL HTSNAME       CHAR(8)  VAR;
  DCL HPART         BIN FIXED(15);
  DCL PART          PIC'ZZ9';
  DCL 1 WORKST,
      2 CREC        CHAR(8)  VAR,
      2 TABC        CHAR(18) VAR,
      2 TSNC        CHAR(8)  VAR,
      2 DBNC        CHAR(8)  VAR,
      2 STO         CHAR(3),
      2 QUE         CHAR(3),
      2 PCARD       PIC'9999999999';

  DCL (SUBSTR,DATE,TIME,NULL,ADDR,LENGTH,INDEX) BUILTIN;
  DCL IC            BIN FIXED(15);
  DCL OUT           CHAR(18) VAR;
```

```
EXEC SQL INCLUDE SQLCA;
IF SUBSTR(PARMS,1,8)=' ' THEN CREC='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,1,8),OUT);
   CREC=OUT;
   IF LENGTH(CREC) < 8 THEN CREC=CREC||'%';
END;
IF SUBSTR(PARMS,9,18)=' ' THEN TABC='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,9,18),OUT);
   TABC=OUT;
   IF LENGTH(TABC) < 18 THEN TABC=TABC||'%';
END;
IF SUBSTR(PARMS,27,8)=' ' THEN TSNC='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,27,8),OUT);
   TSNC=OUT;
   IF LENGTH(TSNC) < 8 THEN TSNC=TSNC||'%';
END;
IF SUBSTR(PARMS,35,8)=' ' THEN DBNC='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,35,8),OUT);
   DBNC=OUT;
   IF LENGTH(DBNC) < 8 THEN DBNC=DBNC||'%';
END;
STO=SUBSTR(PARMS,43,3);
QUE=SUBSTR(PARMS,46,3);
IF SUBSTR(PARMS,49,1Ø) =' '
THEN CARD=-2;
ELSE DO;
     PCARD=SUBSTR(PARMS,49,1Ø);
     CARD=PCARD;
END;

EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
SELECT DISTINCT S.DBNAME, S.NAME, S.PARTITIONS
FROM SYSIBM.SYSTABLES T,
     SYSIBM.SYSTABLESPACE S
WHERE T.CREATOR LIKE :CREC
  AND T.NAME    LIKE :TABC
  AND T.TSNAME  LIKE :TSNC
  AND T.DBNAME  LIKE :DBNC
  AND T.TSNAME  = S.NAME
  AND T.DBNAME  = S.DBNAME
  AND T.CARD > :CARD
  AND T.TYPE = 'T'
ORDER BY 1, 2
FOR FETCH ONLY;
EXEC SQL OPEN C1;

EXEC SQL FETCH C1 INTO :HDBNAME, :HTSNAME, :HPART;
```

```
IF HDBNAME='DSNDBØ4' | HDBNAME='DSNDBØ6' THEN DO;
   PUT SKIP LIST ('ADSNDBØ1  SYSUTILX   1');
   PUT SKIP LIST ('ADSNDBØ1  DBDØ1      1');
   PUT SKIP LIST ('ADSNDBØ1  SCTØ2      1');
   PUT SKIP LIST ('ADSNDBØ1  SPTØ1      1');
   PUT SKIP LIST ('ADSNDBØ1  SYSLGRNX   1');
END;
DO WHILE (SQLCODE=Ø);
   NUMSEQ=1;
   IF HPART=Ø THEN HPART=1;
   PART=HPART;
   PUT SKIP LIST ('A'||HDBNAME||' '||HTSNAME||' '||PART);
   EXEC SQL FETCH C1 INTO :HDBNAME, :HTSNAME, :HPART;
END;
EXEC SQL CLOSE C1;
IF NUMSEQ=Ø THEN DO;
   PUT SKIP LIST ('NO CATALOG ENTRIES FOUND');
   GOTO VEN;
END;

/* SELECTION RESULTS                              */
EXEC SQL DECLARE C2 CURSOR WITH HOLD FOR SELECT
DBNAME, TSNAME, CREATOR, NAME, CARD
FROM SYSIBM.SYSTABLES
WHERE CREATOR LIKE :CREC
  AND NAME    LIKE :TABC
  AND TSNAME  LIKE :TSNC
  AND DBNAME  LIKE :DBNC
  AND CARD > :CARD
  AND TYPE = 'T'
ORDER BY CREATOR,NAME
FOR FETCH ONLY;
EXEC SQL OPEN C2;

EXEC SQL FETCH C2 INTO
     :HDBNAME, :HTSNAME, :HTBCREATOR, :HTBNAME, :HCARD;
IF HDBNAME='DSNDBØ4'|HDBNAME='DSNDBØ6'|HTBCREATOR='SYSIBM' THEN DO;
   PUT SKIP LIST ('BDSNDBØ1  SYSUTILX '||'......... ...... .....');
   PUT SKIP LIST ('BDSNDBØ1  DBDØ1    '||'......... ...... .....');
   PUT SKIP LIST ('BDSNDBØ1  SCTØ2    '||'......... ...... .....');
   PUT SKIP LIST ('BDSNDBØ1  SPTØ1    '||'......... ...... .....');
   PUT SKIP LIST ('BDSNDBØ1  SYSLGRNX '||'......... ...... .....');
END;
DO WHILE (SQLCODE=Ø);
   NUMSEQ=1;
   MCARD=HCARD;
   PUT SKIP LIST ('B'||HDBNAME||' '||HTSNAME||' '||
            SUBSTR(HTBNAME,1,18)||' '||HTBCREATOR||' '||MCARD);
   EXEC SQL FETCH C2 INTO
        :HDBNAME, :HTSNAME, :HTBCREATOR, :HTBNAME, :HCARD;
END;
```

```
  EXEC SQL CLOSE C2;
  IF NUMSEQ=Ø THEN PUT SKIP LIST ('NO CATALOG ENTRIES FOUND');
  FUNC:PROC(INP,OUT);
       DCL INP CHAR(18);
       DCL OUT CHAR(18) VAR;
       DO IC=1 TO 18 BY 1 WHILE (SUBSTR(INP,IC,1) ¬=' ');
       END;
       OUT=SUBSTR(INP,1,IC-1);
   END FUNC;
  VEN:
 END PDBCOPY;
```

PDBMODI

```
* PROCESS GS,OFFSET,OPT(TIME);PDBMODI:PROC(PARMS)OPTIONS(MAIN) REORDER;
/*************************************************************/
/* DESCRIPTION: BUILD MODIFY RECOVERY JOB                    */
/*************************************************************/
  DCL PARMS CHAR(1ØØ) VAR;
  DCL SYSPRINT    FILE STREAM OUTPUT;
  DCL NUMSEQ        BIN FIXED(31) INIT(Ø);
  DCL HDBNAME       CHAR(8)  VAR;
  DCL HTSNAME       CHAR(8)  VAR;
  DCL HTIMESTAMP    CHAR(1Ø);
  DCL AGE           BIN FIXED(31);
  DCL HDSNAME       CHAR(44);
  DCL 1 WORKST,
      2 DBNC        CHAR(8) VAR,
      2 TSNC        CHAR(8) VAR,
      2 DEL         CHAR(1),
      2 DAT         PIC'ZZZZZZZZ';

  DCL (SUBSTR,DATE,TIME,NULL,ADDR,LENGTH,INDEX) BUILTIN;
  DCL IC            BIN FIXED(15);
  DCL OUT           CHAR(18) VAR;
  EXEC SQL INCLUDE SQLCA;
  IF SUBSTR(PARMS,1,8)=' ' THEN DBNC='%';
  ELSE DO;
     CALL FUNC(SUBSTR(PARMS,1,8),OUT);
     DBNC=OUT;
     IF LENGTH(DBNC) < 8 THEN DBNC=DBNC||'%';
  END;
  IF SUBSTR(PARMS,9,8)=' ' THEN TSNC='%';
  ELSE DO;
     CALL FUNC(SUBSTR(PARMS,9,8),OUT);
     TSNC=OUT;
     IF LENGTH(TSNC) < 8 THEN TSNC=TSNC||'%';
  END;
  DEL=SUBSTR(PARMS,17,1);
```

```
DAT=SUBSTR(PARMS,18,8);
AGE=DAT;

IF DEL='D' THEN DO;
   HTIMESTAMP=SUBSTR(DAT,1,4)||'-'||SUBSTR(DAT,5,2)||'-'||
              SUBSTR(DAT,7,2);
   EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
   SELECT DBNAME, TSNAME, DATE(TIMESTAMP), DSNAME
   FROM SYSIBM.SYSCOPY
   WHERE DBNAME LIKE :DBNC
     AND TSNAME LIKE :TSNC
     AND DATE(TIMESTAMP) < :HTIMESTAMP
     AND ICTYPE = 'F'
   ORDER BY DBNAME, TSNAME, 3
   FOR FETCH ONLY;

   EXEC SQL OPEN C1;

   EXEC SQL FETCH C1 INTO :HDBNAME,:HTSNAME,:HTIMESTAMP,:HDSNAME;
   DO WHILE (SQLCODE=Ø);
      NUMSEQ=1;
      PUT SKIP LIST
      (HDBNAME||' '||HTSNAME||' '||HTIMESTAMP||' '||HDSNAME);
      EXEC SQL FETCH C1 INTO :HDBNAME,:HTSNAME,:HTIMESTAMP,:HDSNAME;
   END;
   EXEC SQL CLOSE C1;
   IF NUMSEQ=Ø THEN DO;
      PUT SKIP LIST ('NO CATALOG ENTRIES FOUND');
      GOTO VEN;
   END;
END;

IF DEL='A' THEN DO;
   EXEC SQL DECLARE C2 CURSOR WITH HOLD FOR
   SELECT DBNAME, TSNAME, DATE(TIMESTAMP), DSNAME
   FROM SYSIBM.SYSCOPY
   WHERE DBNAME LIKE :DBNC
     AND TSNAME LIKE :TSNC
     AND DATE(TIMESTAMP) < CHAR(CURRENT DATE - :AGE DAYS)
     AND ICTYPE = 'F'
   ORDER BY DBNAME, TSNAME, 3
   FOR FETCH ONLY;

   EXEC SQL OPEN C2;

   EXEC SQL FETCH C2 INTO :HDBNAME,:HTSNAME,:HTIMESTAMP,:HDSNAME;
   DO WHILE (SQLCODE=Ø);
      NUMSEQ=1;
      PUT SKIP LIST
      (HDBNAME||' '||HTSNAME||' '||HTIMESTAMP||' '||HDSNAME);
      EXEC SQL FETCH C2 INTO :HDBNAME,:HTSNAME,:HTIMESTAMP,:HDSNAME;
      END;
      EXEC SQL CLOSE C2;
      IF NUMSEQ=Ø THEN DO;
         PUT SKIP LIST ('NO CATALOG ENTRIES FOUND');
         GOTO VEN;
      END;
   END;

   FUNC:PROC(INP,OUT);
        DCL INP CHAR(18);
        DCL OUT CHAR(18) VAR;
        DO IC=1 TO 18 BY 1 WHILE (SUBSTR(INP,IC,1) ¬=' ');
        END;
        OUT=SUBSTR(INP,1,IC-1);
    END FUNC;
   VEN:
  END PDBMODI;
```

PDBRECO

```
* PROCESS GS,OFFSET,OPT(TIME);PDBRECO:PROC(PARMS)OPTIONS(MAIN) REORDER;
/*****************************************************************/
/* DESCRIPTION: PL/I PROGRAM FOR RECOVER UTILITY                 */
/*****************************************************************/
  DCL PARMS CHAR(1ØØ) VAR;
  DCL SYSPRINT    FILE STREAM OUTPUT;
  DCL NUMSEQ        BIN FIXED(31) INIT(Ø);
  DCL MCARD         PIC'—.—.—.—9';
  DCL HCARD         BIN FIXED(31);
  DCL HTBNAME       CHAR(18);
  DCL HTBCREATOR    CHAR(8);
  DCL HDBNAME       CHAR(8)  VAR;
  DCL HTSNAME       CHAR(8)  VAR;
  DCL HPART         BIN FIXED(15);
  DCL PART          PIC'ZZ9';
  DCL DBNAME        CHAR(8);
  DCL TSNAME        CHAR(8);
  DCL DSNUM         BIN FIXED(31);
  DCL ICTYPE        CHAR(1);
  DCL ICDATE        CHAR(1Ø);
  DCL ICTIME        CHAR(8);
  DCL START_RBA     CHAR(12);
  DCL DSNAME        CHAR(44);
  DCL 1 WORKST,
      2 CREC        CHAR(8)  VAR,
      2 TABC        CHAR(18) VAR,
      2 TSNC        CHAR(8)  VAR,
      2 DBNC        CHAR(8)  VAR,
      2 RTYP        CHAR(6),
      2 DFR         CHAR(8),
      2 DTO         CHAR(8);

  DCL (DATEFROM, DATETO) CHAR(1Ø);

  DCL (SUBSTR,DATE,TIME,NULL,ADDR,LENGTH,INDEX) BUILTIN;
  DCL IC            BIN FIXED(15);
  DCL OUT           CHAR(18) VAR;
  EXEC SQL INCLUDE SQLCA;
  IF SUBSTR(PARMS,1,8)=' ' THEN CREC='%';
  ELSE DO;
     CALL FUNC(SUBSTR(PARMS,1,8),OUT);
     CREC=OUT;
     IF LENGTH(CREC) < 8 THEN CREC=CREC||'%';
  END;
  IF SUBSTR(PARMS,9,18)=' ' THEN TABC='%';
  ELSE DO;
     CALL FUNC(SUBSTR(PARMS,9,18),OUT);
     TABC=OUT;
     IF LENGTH(TABC) < 18 THEN TABC=TABC||'%';
  END;
  IF SUBSTR(PARMS,27,8)=' ' THEN TSNC='%';
  ELSE DO;
     CALL FUNC(SUBSTR(PARMS,27,8),OUT);
     TSNC=OUT;
     IF LENGTH(TSNC) < 8 THEN TSNC=TSNC||'%';
  END;
  IF SUBSTR(PARMS,35,8)=' ' THEN DBNC='%';
  ELSE DO;
     CALL FUNC(SUBSTR(PARMS,35,8),OUT);
     DBNC=OUT;
     IF LENGTH(DBNC) < 8 THEN DBNC=DBNC||'%';
  END;
  RTYP=SUBSTR(PARMS,43,6);
  DFR =SUBSTR(PARMS,49,8);
  DTO =SUBSTR(PARMS,57,8);
  DATEFROM=SUBSTR(DFR,1,4)||'-'||SUBSTR(DFR,5,2)||'-'||SUBSTR(DFR,7,2);
  DATETO  =SUBSTR(DTO,1,4)||'-'||SUBSTR(DTO,5,2)||'-'||SUBSTR(DTO,7,2);

  EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
  SELECT DISTINCT S.DBNAME, S.NAME, S.PARTITIONS
  FROM SYSIBM.SYSTABLES T,
       SYSIBM.SYSTABLESPACE S
  WHERE T.CREATOR LIKE :CREC
    AND T.NAME    LIKE :TABC
    AND T.TSNAME  LIKE :TSNC
    AND T.DBNAME  LIKE :DBNC
    AND T.TSNAME  = S.NAME
    AND T.DBNAME  = S.DBNAME
    AND T.TYPE = 'T'
  ORDER BY 1, 2
  FOR FETCH ONLY;
  EXEC SQL OPEN C1;

  EXEC SQL FETCH C1 INTO :HDBNAME, :HTSNAME, :HPART;
  DO WHILE (SQLCODE=Ø);
     NUMSEQ=1;
     IF HPART=Ø THEN HPART=1;
     PART=HPART;
     PUT SKIP LIST ('A'||HDBNAME||' '||HTSNAME||' '||PART);
     EXEC SQL FETCH C1 INTO :HDBNAME, :HTSNAME, :HPART;
  END;
  EXEC SQL CLOSE C1;
  IF NUMSEQ=Ø THEN DO;
     PUT SKIP LIST ('NO CATALOG ENTRIES FOUND');
     GOTO VEN;
  END;

  /* SELECTION RESULTS                              */
  EXEC SQL DECLARE C11 CURSOR WITH HOLD FOR SELECT
  DBNAME, TSNAME, CREATOR, NAME, CARD
  FROM SYSIBM.SYSTABLES
  WHERE CREATOR LIKE :CREC
    AND NAME    LIKE :TABC
    AND TSNAME  LIKE :TSNC
    AND DBNAME  LIKE :DBNC
    AND TYPE = 'T'
  ORDER BY CREATOR,NAME
  FOR FETCH ONLY;
  EXEC SQL OPEN C11;
  EXEC SQL FETCH C11 INTO
       :HDBNAME, :HTSNAME, :HTBCREATOR, :HTBNAME, :HCARD;
  DO WHILE (SQLCODE=Ø);
     NUMSEQ=1;
     MCARD=HCARD;
     PUT SKIP LIST ('B'||HDBNAME||' '||HTSNAME||SUBSTR(HTBNAME,1,18)
                       ||' '||HTBCREATOR||' '||MCARD);
     EXEC SQL FETCH C11 INTO
          :HDBNAME, :HTSNAME, :HTBCREATOR, :HTBNAME, :HCARD;
  END;
  EXEC SQL CLOSE C11;

  /* SELECTION FOR TOCOPY OPTION                        */
  IF RTYP='TOCOPY' THEN DO;
     EXEC SQL DECLARE C2 CURSOR WITH HOLD FOR
     SELECT DISTINCT C.DBNAME,C.TSNAME,DSNUM,ICTYPE,
            DATE(TIMESTAMP),TIME(TIMESTAMP),DSNAME
     FROM SYSIBM.SYSCOPY C,
          SYSIBM.SYSTABLES T
     WHERE T.CREATOR LIKE :CREC
       AND T.NAME    LIKE :TABC
       AND T.TSNAME  LIKE :TSNC
       AND T.DBNAME  LIKE :DBNC
       AND T.TYPE = 'T'
       AND C.DBNAME = T.DBNAME
       AND C.TSNAME = T.TSNAME
       AND DATE(TIMESTAMP) BETWEEN :DATEFROM AND :DATETO
       AND C.ICTYPE = 'F'
     ORDER BY DBNAME, TSNAME, DSNUM, 5 DESC, 6
     FOR FETCH ONLY;

     EXEC SQL OPEN C2;

     EXEC SQL FETCH C2 INTO
     :DBNAME, :TSNAME, :DSNUM, :ICTYPE, :ICDATE, :ICTIME, :DSNAME;
     DO WHILE (SQLCODE=Ø);
        NUMSEQ=1;
        PUT SKIP LIST ('C'||DBNAME||' '||TSNAME||' '||DSNUM||' '||
                      ICTYPE||' '||ICDATE||' '||ICTIME||' '||DSNAME);
        EXEC SQL FETCH C2 INTO
        :DBNAME, :TSNAME, :DSNUM, :ICTYPE, :ICDATE, :ICTIME, :DSNAME;
     END;
     EXEC SQL CLOSE C2;
  END;
  /* SELECTION FOR RBA OPTION                            */
  IF RTYP='RBA' THEN DO;
     EXEC SQL DECLARE C3 CURSOR WITH HOLD FOR
     SELECT DISTINCT C.DBNAME,C.TSNAME,DSNUM,ICTYPE,
            DATE(TIMESTAMP),TIME(TIMESTAMP),HEX(START_RBA)
     FROM SYSIBM.SYSCOPY C,
          SYSIBM.SYSTABLES T
     WHERE T.CREATOR LIKE :CREC
       AND T.NAME    LIKE :TABC
       AND T.TSNAME  LIKE :TSNC
       AND T.DBNAME  LIKE :DBNC
       AND T.TYPE = 'T'
       AND C.ICTYPE NOT IN ('T')
       AND C.ICBACKUP=''
       AND C.DBNAME = T.DBNAME
       AND C.TSNAME = T.TSNAME
       AND DATE(TIMESTAMP) BETWEEN :DATEFROM AND :DATETO
     ORDER BY DBNAME, TSNAME, DSNUM, 5 DESC, 6
     FOR FETCH ONLY;

     EXEC SQL OPEN C3;

     EXEC SQL FETCH C3 INTO
     :DBNAME, :TSNAME, :DSNUM, :ICTYPE, :ICDATE, :ICTIME, :START_RBA;
     DO WHILE (SQLCODE=Ø);
```

```
        NUMSEQ=1;
        PUT SKIP LIST ('C'||DBNAME||' '||TSNAME||' '||DSNUM||' '||
                 ICTYPE||' '||ICDATE||' '||ICTIME||' '||START_RBA);
        EXEC SQL FETCH C3 INTO
        :DBNAME,:TSNAME,:DSNUM,:ICTYPE,:ICDATE,:ICTIME,:START_RBA;
     END;
     EXEC SQL CLOSE C3;
  END;
  IF NUMSEQ=Ø THEN DO;
     PUT SKIP LIST ('NO CATALOG ENTRIES FOUND');
     GOTO VEN;
  END;
  FUNC:PROC(INP,OUT);
       DCL INP CHAR(18);
       DCL OUT CHAR(18) VAR;
       DO IC=1 TO 18 BY 1 WHILE (SUBSTR(INP,IC,1) ¬=' ');
       END;
       OUT=SUBSTR(INP,1,IC-1);
   END FUNC;
  VEN:
 END PDBRECO;
```

PDBSTOS

```
* PROCESS GS,OFFSET,OPT(TIME);PDBSTOS:PROC(PARMS)OPTIONS(MAIN) REORDER;
/*****************************************************************/
/* DESCRIPTION: BUILD STOSPACE JOB                               */
/*****************************************************************/
  DCL PARMS CHAR(1ØØ) VAR;
  DCL SYSPRINT    FILE STREAM OUTPUT;
  DCL NUMSEQ        BIN FIXED(31) INIT(Ø);
  DCL CREATOR       CHAR(8)  VAR;
  DCL SGNAME        CHAR(8)  VAR;
  DCL 1 WORKST,
      2 CRE         CHAR(8) VAR,
      2 SGN         CHAR(8) VAR,
      2 SGA         CHAR(3);

  DCL (SUBSTR,DATE,TIME,NULL,ADDR,LENGTH,INDEX) BUILTIN;
  DCL IC            BIN FIXED(15);
  DCL OUT           CHAR(18) VAR;
  EXEC SQL INCLUDE SQLCA;
  IF SUBSTR(PARMS,1,8)=' ' THEN CRE='%';
  ELSE DO;
     CALL FUNC(SUBSTR(PARMS,1,8),OUT);
     CRE=OUT;
     IF LENGTH(CRE) < 8 THEN CRE=CRE||'%';
  END;
  IF SUBSTR(PARMS,9,8)=' ' THEN SGN='%';
  ELSE DO;
     CALL FUNC(SUBSTR(PARMS,9,8),OUT);
     SGN=OUT;
     IF LENGTH(SGN) < 8 THEN SGN=SGN||'%';
  END;
  SGA=SUBSTR(PARMS,17,3);

  IF SGA='NO' THEN DO;
     EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
     SELECT CREATOR, NAME
     FROM SYSIBM.SYSSTOGROUP
     WHERE CREATOR LIKE :CRE
       AND NAME    LIKE :SGN
     ORDER BY CREATOR, NAME
     FOR FETCH ONLY;

     EXEC SQL OPEN C1;

     EXEC SQL FETCH C1 INTO :CREATOR, :SGNAME;
     DO WHILE (SQLCODE=Ø);
        NUMSEQ=1;
        PUT SKIP LIST (CREATOR||' '||SGNAME);
        EXEC SQL FETCH C1 INTO :CREATOR, :SGNAME;
     END;
     EXEC SQL CLOSE C1;
     IF NUMSEQ=Ø THEN DO;
        PUT SKIP LIST ('NO CATALOG ENTRIES FOUND');
        GOTO VEN;
     END;
  END;
  ELSE PUT SKIP LIST ('-        ALL');

  FUNC:PROC(INP,OUT);
       DCL INP CHAR(18);
       DCL OUT CHAR(18) VAR;
       DO IC=1 TO 18 BY 1 WHILE (SUBSTR(INP,IC,1) ¬=' ');
       END;
       OUT=SUBSTR(INP,1,IC-1);
   END FUNC;
  VEN:
 END PDBSTOS;
```

DBUT00

```
DBUTØØ1          .ALARM = YES  .WINDOW=NORESP .ALARM = YES
'&message'
```

Bernard Zver
Database Administrator
Informatika Maribor (Slovenia)

ALTERing DSETPASS using DSNTEP2

A planning requirement for DB2 UDB for OS/390 Version 6 is to remove passwords from indexes and tablespaces prior to the migration. IBM sample program DSNTEP2 can be used to create the ALTER statements and remove the passwords of objects using DB2 dataset protection.

A SELECT can be issued from DSNTEP2 to produce rows that have the appropriate syntax for ALTERing DSETPASS to remove the passwords. The DSNTEP2 SYSPRINT output can then be sorted to remove extraneous report lines and the sorted records can be used directly as input into DSNTEP2, in a subsequent step to ALTER the passwords to blanks.

Please note that the sort syntax in the sample provided depends on several factors such as the software vendor used, and the offset of literal strings (eg ‘ALTER’) in the DSNTEP2 SYSPRINT output report.

Figure 1 shows a subset of the sample output from the IEBGENR1 step (before the sort).

Figure 2 shows a subset of the sample output from the IEBGENR2 step (after the sort).

```
.
.
13_≥ ALTER TABLESPACE DSN8D51A.DSN8S51S DSETPASS " ";
14_≥ ALTER TABLESPACE DSN8D51P.DSN8S51C DSETPASS " ";
15_≥ ALTER TABLESPACE DSN8D51P.DSN8S51Q DSETPASS " ";
16_≥ ALTER INDEX DSN831Ø.XEMP1 DSETPASS " ";
17_≥ ALTER INDEX DSN831Ø.XEMP2 DSETPASS " ";
18_≥ ALTER INDEX DSN831Ø.XCONA1 DSETPASS " ";
  .
  .
```

Figure 1: Subset of sample output from the IEBGENR1 step

```
 .
 .
ALTER INDEX DSN851Ø.XPARTS DSETPASS " ";
ALTER INDEX DSN851Ø.XPROJAC1 DSETPASS " ";
ALTER INDEX DSN851Ø.XPROJ1 DSETPASS " ";
ALTER INDEX DSN851Ø.XPROJ2 DSETPASS " ";
ALTER TABLESPACE DSN8D23A.DSN8S23D DSETPASS " ";
ALTER TABLESPACE DSN8D23A.DSN8S23E DSETPASS " ";
ALTER TABLESPACE DSN8D23A.DSN8S23R DSETPASS " ";
ALTER TABLESPACE DSN8D23P.DSN8S23C DSETPASS " ";
 .
 .
 .
```

Figure 2: Subset of sample output from the IEBGENR2 step

The DML to generate, sort, and execute ALTER statements for DSETPASS follows:

```
//*————————————————————————————————————————*
//* IDENTIFY TS/IX OBJECTS THAT HAVE A NON-BLANK VALUE FOR DSETPASS AND
//* BUILD ALTER STATEMENTS TO SET THE DSETPASS VALUE TO BLANKS.
//*————————————————————————————————————————*
//IDENTFY  EXEC PGM=IKJEFTØ1
//STEPLIB  DD DISP=SHR,DSN=SDC.OPLIB.DB2X.LOADLIB
//SYSPRINT DD DSN=&&DSETIN,UNIT=SYSDA,
//            DISP=(NEW,PASS,DELETE),SPACE=(TRK,(15,5),RLSE)
//ABNLIGNR DD DUMMY
//SYSTERM  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(DB2X)
   RUN PROGRAM(DSNTEP2) PLAN(DSNTEP2)
   END
/*
//SYSIN    DD *
 SELECT

 'ALTER TABLESPACE ' CONCAT

   STRIP(DBNAME,TRAILING) CONCAT '.' CONCAT STRIP(NAME,TRAILING) CONCAT

   ' DSETPASS ' CONCAT '" ' CONCAT '"' CONCAT ';' CONCAT
```

```
    '                              '


      FROM SYSIBM.SYSTABLESPACE
         WHERE DSETPASS  <> '        '
 UNION ALL
 SELECT

 'ALTER INDEX ' CONCAT

   STRIP(CREATOR,TRAILING) CONCAT '.' CONCAT STRIP(NAME,TRAILING) CONCAT

   ' DSETPASS ' CONCAT '" ' CONCAT '"' CONCAT ';' CONCAT

    '                              '


      FROM SYSIBM.SYSINDEXES
         WHERE DSETPASS  <> '        ';
/*
//*-----------------------------------------------------------------*
//* IEBGENER THE "SELECT" OUTPUT TO HAVE AN AUDIT TRAIL OF THE ROWS
//*-----------------------------------------------------------------*
//IEBGENR1 EXEC PGM=IEBGENER,COND=(Ø,NE)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1   DD DSN=&&DSETIN,DISP=(OLD,PASS,DELETE)
//SYSUT2   DD SYSOUT=*
//*
//*-----------------------------------------------------------------*
//* SORT THE ALTER STATEMENTS (SYNCSORT).
//*
//* STRIP EXTRANEOUS LINES FROM THE DSNTEP2 SYSPRINT OUTPUT.
//* THE DSNTEP2 OUTPUT REPORT HAS THE LITERAL 'ALTER'
//* STARTING IN COLUMN 29 (FOR A LENGTH OF 5 BYTES).
//* THE 'INREC' SPECIFICATION REFORMATS THE INPUT RECORD BEFORE THE
//* SORT. IT IS A 72-BYTE RECORD (FOR DSNTEP2 INPUT).
//* THE 'INCLUDE' SELECTS ONLY FROM THE COMPARISON GIVEN, HENCE
//* ONLY THE DATA IS SELECTED AND THE EXTRANEOUS LINES FROM THE REPORT
//* (EG LINES WITH DASHES, LINES WITH SPACES, ETC) ARE NOT SELECTED.
//* SINCE "OUTFIL' IS SPECIFIED, THE OUTPUT DD OF SORTOFx IS USED.
//*-----------------------------------------------------------------*
//SRTPASS1 EXEC PGM=SYNCSORT,COND=(Ø,NE)
//SYSOUT   DD SYSOUT=A
//SORTWKØ1 DD UNIT=SYSDA,SPACE=(CYL,(1))
//SYSIN    DD *
  INREC FIELDS=(29,72)
  SORT  FIELDS=(1,72,CH,A)
  OUTFIL FILES=1,
   INCLUDE=(1,5,CH,EQ,C'ALTER'),
   OUTREC=(1,72)
  END
/*
//SORTIN   DD DISP=(OLD,DELETE,DELETE),DSN=&&DSETIN
//SORTOF1  DD DSN=&&DSETOUT,
//            UNIT=SYSDA,
//            DCB=(RECFM=FB,LRECL=72,BLKSIZE=27ØØØ),
//            SPACE=(TRK,(1,1),RLSE),
//            DISP=(NEW,PASS,DELETE)
//*-------------------------------------*
//* IEBGENER THE ALTER STATEMENT CARDS TO HAVE AN AUDIT TRAIL
//*-------------------------------------*
//IEBGENR2 EXEC PGM=IEBGENER,COND=(Ø,NE)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1   DD DSN=&&DSETOUT,DISP=(OLD,PASS,DELETE)
//SYSUT2   DD SYSOUT=*
//*
//*-------------------------------------*
//* ALTER THE DSETPASS TO BLANKS FOR TABLESPACES AND INDEXES
//*-------------------------------------*
//ALTERPW  EXEC PGM=IKJEFTØ1,COND=(Ø,NE)
//STEPLIB  DD DISP=SHR,DSN=SDC.OPLIB.DB2X.LOADLIB
//ABNLIGNR DD DUMMY
//SYSTERM  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(DB2X)
   RUN PROGRAM(DSNTEP2) PLAN(DSNTEP2)
   END
/*
//SYSIN    DD DISP=(OLD,DELETE,DELETE),DSN=&&DSETOUT
/*
//
```

John Mustric
Senior System Programmer
Columbia Energy Group (USA)

DB2 Update is looking for REXX EXECs, macros, program code, etc, that experienced DB2 users have written to make their life easier. We will publish them (after vetting by our expert panel) and send you a cheque when the article is published. Articles can be of any length and can be sent or e-mailed to Robert Burgess at any of the addresses shown on page 2.

Triggers and DB2 Version 6

Most relational database management systems provide support for triggers, and IBM is adding trigger support to DB2 for OS/390 in Version 6 (to be generally available in June 1999). But just what is a trigger? If you have never had the opportunity to use them, their power may elude you at first. However, once you have used triggers, living without them can be unthinkable!

THE BASICS

Simply stated, a trigger is a piece of code that is executed in response to a data modification statement; that is, an insert, update, or delete. To be a bit more precise, triggers are event-driven specialized procedures that are stored in, and managed by, the RDBMS. Each trigger is attached to a single specified table.

Triggers can be thought of as an advanced form of ‘rule’ or ‘constraint’ written using an extended form of SQL. A trigger cannot be directly called or executed; it is automatically executed (or ‘fired’) by the RDBMS as the result of an action – a data modification to the associated table.

Once a trigger is created it is always executed when its ‘firing’ event occurs (update, insert, or delete). Therefore, triggers are automatic, implicit, and non-bypassable.

TRIGGERS VERSUS STORED PROCEDURES

Triggers are also similar to stored procedures. Both consist of procedural logic that is stored at the database level. However, stored procedures are not event-driven and are not attached to a specific table.

A stored procedure is explicitly executed by invoking a CALL to the procedure (instead of implicitly being executed like triggers). Additionally, a stored procedure can access many tables without being specifically associated with any of them.

DB2 has supported stored procedures since Version 4.

WHY USE TRIGGERS?

Triggers are useful for implementing code that must be executed on a regular basis in response to a pre-defined event. By utilizing triggers, scheduling and data integrity problems can be eliminated because the trigger will be fired whenever the triggering event occurs. You need not remember to schedule or code an activity to perform the logic in the trigger. It happens automatically by virtue of it being in the trigger. This is true of both static and dynamic SQL (*ad hoc* and planned SQL).

Triggers can be implemented for many practical uses. Quite often it is impossible to code business rules into the database using only DDL. For example, DB2 does not support complex constraints (only value-based CHECK constraints) or various types of referential constraints (such as pendant DELETE processing or ON UPDATE CASCADE).

Using triggers, a very flexible environment is established for implementing business rules and constraints in the DBMS. This is important because having the business rules in the database ensures that everyone uses the same logic to accomplish the same process.

Triggers can be coded to access and/or modify other tables, print informational messages, and specify complex restrictions. For example, consider the standard suppliers and parts application used in most introductory database texts. A part can be supplied by many suppliers and a supplier can supply many parts. Triggers can be used to support the following scenarios:

- What if a business rule exists specifying that no more than three suppliers are permitted to supply any single part? A trigger can be coded to check that rows cannot be inserted if the data violates this requirement.
- A trigger can be created to allow orders only for parts that are already in stock; or, maybe for parts that are already in stock or are on order and planned for availability within the next week.
- Triggers can be used to perform calculations such as ensuring that the order amount for the parts is calculated appropriately, given the suppliers chosen to provide the parts. This is especially useful if the order purchase amount is stored in the database as redundant data.

- To curb costs – a business decision may be made that the low cost supplier will always be used. A trigger can be implemented to disallow any order that is not the current ‘lowest cost’ order.

The number of business rules that can be implemented using triggers is truly limited only by your imagination (or, more appropriately, by your business needs).

Additionally, triggers can access non-DB2 resources. This can be accomplished by invoking a stored procedure or a user-defined function that takes advantage of the OS/390 Resource Recovery Services (RRS). Data stored in the non-DB2 resource can be accessed or modified in the stored procedure or user-defined function that is called.

WHEN DOES A TRIGGER FIRE?

Two options exist for when a trigger can fire: before the firing activity occurs or after the firing activity occurs. DB2 supports both ‘before’ and ‘after’ triggers. A ‘before’ trigger executes before the firing activity occurs; an ‘after’ trigger executes after the firing activity occurs. In DB2 Version 6, ‘before’ triggers are restricted because they cannot perform updates.

Knowing how the triggers in your database function is imperative. Without this knowledge, properly functioning triggers cannot be coded, supported, or maintained effectively.

Consider, for example, if the firing activity occurs before the trigger is fired. In other words, the update, insert, or delete occurs first – as a result of this action, the trigger logic is executed. If necessary, the trigger code can ‘roll back’ the data modification.

What if the trigger is fired before the actual firing event occurs? In this situation a roll back would not be required for the firing event code because it has not yet occurred. However, a roll back may be required for any data modifications that occurred prior to this firing event within the same transaction.

Another interesting feature of DB2 Version 6 triggers is the order in which they are fired. If multiple triggers are coded on the same table, which trigger is fired first? It can make a difference as to how the triggers should be coded, tested, and maintained.

The rule for order of execution is basically simple to understand, but can be difficult to maintain. For triggers of the same type, they are executed in the order in which they were created. For example, if two ‘delete’ triggers are coded on the same table, the one that was physically created first is executed first. Keep this in mind as you make changes to your database. If you need to drop the table and re-create it to implement a schema change, make sure you create the triggers in the desired (same) order to keep the functionality the same.

As can readily be seen, determining the procedural activity that is required when triggers are present can be a complicated task. It is of paramount importance that all developers are schooled in the firing methods utilized for triggers in DB2 Version 6.

TRIGGER PACKAGES

When a trigger is executed, DB2 creates a trigger package for the statements in the triggered action. The trigger package is recorded in SYSIBM.SYSPACKAGE and has the same name as the trigger. The trigger package is always accessible and can be executed only when a trigger is activated by a triggering operation.

To delete the trigger package, you must use the DROP TRIGGER statement.

TRIGGERS CAN FIRE OTHER TRIGGERS

As we’ve already learned, a trigger is fired by an insert, update, or delete. However, a trigger can also contain insert, update, and delete logic within itself.

Therefore, a trigger is fired by a data modification, but can also cause another data modification, thereby firing yet another trigger. When a trigger contains insert, update, and/or delete logic, the trigger is said to be a nested trigger.

Most DBMSs, however, place a limit on the number of nested triggers that can be executed within a single firing event. If this were not done, it could be quite possible to have triggers firing triggers *ad infinitum* until all of the data was removed from an entire database!

If referential integrity is combined with triggers, additional cascading updates and/or deletes can occur. If a delete or update results in a series of additional updates or deletes that need to be propagated to other tables, then the update or delete triggers for the second table also will be activated.

This combination of multiple triggers and referential integrity constraints are capable of setting a cascading effect into motion, which can result in multiple data changes. DB2 Version 6 limits this cascading effect to 16 levels in order to prevent endless looping. If more than 16 levels of nesting occur, the transaction is aborted.

The ability to nest triggers provides an efficient method for implementing automatic data integrity. Because triggers generally cannot be by-passed, they provide an elegant solution to the enforced application of business rules. Use caution, however, to ensure that the maximum trigger nesting level is not reached. Failure to heed this advice can cause an environment where certain types of update cannot occur!

TRIGGER LOCATIONS

There are limits to what triggers can accomplish. As of DB2 Version 6, you cannot define triggers on:

- A system catalog table
- PLAN_TABLE
- STATEMENT_TABLE
- DSN_FUNCTION_TABLE
- View
- Alias
- Synonym

- Any table with a three-part name.

USING TRIGGERS TO IMPLEMENT REFERENTIAL INTEGRITY

One of the primary uses for triggers is to support Referential Integrity (RI). Although DB2 supports a very robust form of declarative RI, no current DBMS fully supports all possible referential constraints. This is true of DB2, as well. A listing of these possibilities is given below:

- DELETE RESTRICT – if any rows exist in the dependent table, the primary key row in the parent table cannot be deleted.
- DELETE CASCADE – if any rows exist in the dependent table, the primary key row in the parent table is deleted, and all dependent rows are also deleted.
- DELETE NEUTRALIZE – if any rows exist in the dependent table, the primary key row in the parent table is deleted, and the foreign key column(s) for all dependent rows are set to NULL as well.
- UPDATE RESTRICT – if any rows exist in the dependent table, the primary key column(s) in the parent table cannot be updated.
- UPDATE CASCADE – if any rows exist in the dependent table, the primary key column(s) in the parent table are updated, and all foreign key values in the dependent rows are updated to the same value.
- UPDATE NEUTRALIZE – if any rows exist in the dependent table, the primary key row in the parent table is deleted, and all foreign key values in the dependent rows are updated to NULL as well.
- INSERT RESTRICT – a foreign key value cannot be inserted into the dependent table unless a primary key value already exists in the parent table.
- FK UPDATE RESTRICTION – a foreign key cannot be updated to a value that does not already exist as a primary key value in the parent table.

- PENDANT DELETE – when the last foreign key value in the dependent table is deleted, the primary key row in the parent table is also deleted.

Triggers can be coded, in lieu of declarative RI, to support all of the RI rules shown above. Of course, when you use triggers, it necessitates writing procedural code for each rule for each constraint, whereas declarative RI constraints are coded in the DDL that is used to create relational tables.

In order to use triggers to support RI rules, it is sometimes necessary to know the values impacted by the action that fired the trigger. For example, consider the case where a trigger is fired because a row was deleted. The row, and all of its values, has already been deleted because the trigger is executed after its firing action occurs. But if this is the case, how can we ascertain whether referentially connected rows exist with those values?

The solution is provided in the form of two specialized aliases available only inside triggers – NEW and OLD.

Each trigger can have one NEW view of the table and one OLD view of the table available. Once again, these 'views' are accessible only from triggers. They provide access to the modified data by viewing information in the transaction log. The transaction log is a record of all data modification activity, automatically maintained by the DBMS.

Figure 1 shows before and after views of table activity.

When an INSERT occurs, the NEW table contains the rows that were just inserted into the table to which the trigger is attached. When a DELETE occurs, the OLD table contains the rows that were just deleted from the table to which the trigger is attached. An UPDATE statement logically functions as a DELETE followed by an INSERT. Therefore, after an UPDATE, the NEW table contains the new values for the rows that were just updated in the table to which the trigger is attached; the OLD table contains the old values for the updated rows.

Therefore, the trigger can use these specialized NEW and OLD table views to query the affected data. Remember, too, that SQL data modification can occur a set at a time. One DELETE or UPDATE

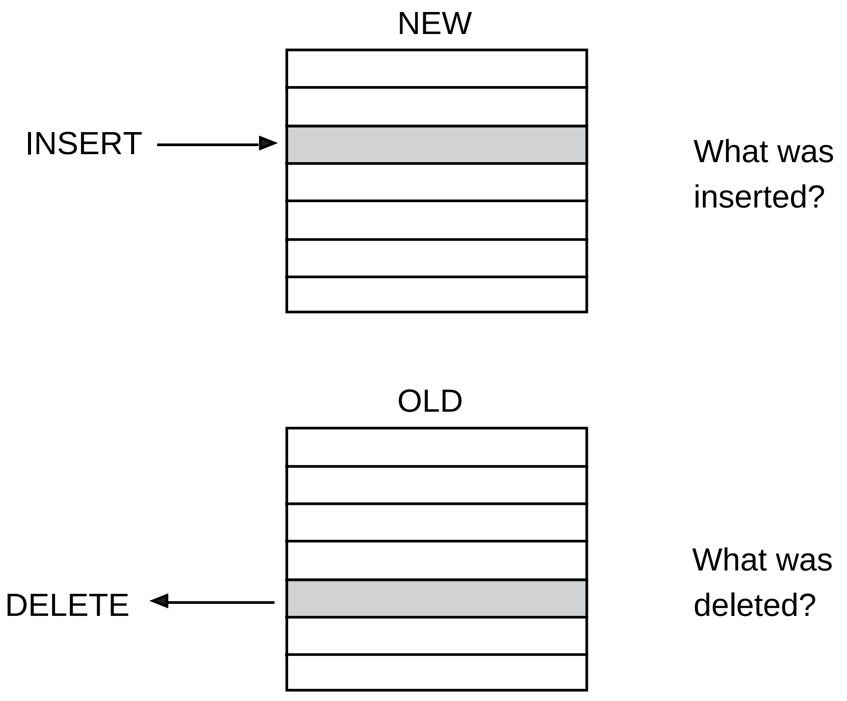

Figure 1: Before and after views of table activity

statement can impact multiple rows. This must be taken into account when coding the actual trigger logic.

Additionally, the alias names, OLD and NEW, can be changed if so desired (for example, to INSERTED and DELETED, the names used by SQL Server).

TRIGGER GRANULARITY

Because SQL is a set-level language, any single SQL statement can impact multiple rows of data. For example, one DELETE statement can actually cause zero, one, or many rows to be removed. Triggers need to take this into account.

Therefore, there are two levels of granularity that a trigger can have: statement level or row level. A statement-level trigger is executed once upon firing, regardless of the actual number of rows inserted, deleted, or updated. A row-level trigger, once fired, is executed once for each and every row that is inserted, deleted, or updated.

Different business requirements will drive what type of trigger granularity should be chosen.

A SAMPLE TRIGGER

Browse the sample trigger shown below. It is an update trigger, coded on the EMP table. This trigger implements a simple check to ensure that pay rises are less than 50%. When the new salary exceeds 50% of the prior salary, an error is raised.

```
CREATE TRIGGER SALARY_UPDATE
  BEFORE UPDATE OF SALARY
  ON EMP
  FOR EACH ROW MODE DB2SQL

WHEN (NEW.SALARY > (OLD.SALARY * 1.5))
BEGIN ATOMIC
  SIGNAL SQLSTATE '75001' ('Rise exceeds 50%');
END;
```

The trigger executes once for each row. So if multiple rows are modified by a single update, the trigger will run multiple times, once for each row modified. Also, the trigger will be run BEFORE the actual modification occurs. Finally, take special notice how NEW and OLD are used to check values before and after the update.

SYNOPSIS

Triggers are a powerful feature; they enable non-bypassable event-driven logic to be intrinsically intermingled with data. As of Version 6, DB2 for OS/390 will support triggers. It is a wise course of action to learn about triggers and what they can provide, so that you can benefit from implementing triggers after you migrate to DB2 Version 6.

Craig S Mullins
VP, Operations
PLATINUM Technology (USA)

DB2 news

DB2 users can benefit from Princeton Softech's Move for Servers data extract, transformation, and migration tool. Users can create and exchange relationally-intact data subsets between DB2 Universal Database, DB2 for OS/390, and Oracle.

Move for Servers includes functions for scheduling, bulk loading, and relational deletes. It also supports ageing features, specific to the needs of Y2K projects, including object-level semantic ageing, date interpretation, semantic safeguard, target dating, global date advancement, window watching, date absorption, and date skipping.

For further information contact:
Princeton Softech, 1060 State Road, Princeton, NJ 08540-1423, USA.
Tel: (609) 497 0205.
URL: http://www.princetonsoftech.com.

* * *

IBM has announced Version 2.4 of its DB2 Digital Library for storage, management, and distribution of digital content. It comprises a core Library Server, one or more multimedia Object Servers, and one or more clients.

The Library Server, which manages catalogue information, locates stored objects using a variety of search technologies, provides access to the objects held in the collection, and communicates with distributed Object Servers, which store the digital content. Client applications get direct access to the information requested from the collection, regardless of where the data is stored.

The Digital Library architecture, which is implemented across multiple platforms, supports multiple distributed Object Servers and the logical separation of applications, indices, and data. Key features of the library include advanced search facilities, extensive and customizable access control, access via the Internet, and scalability from a single workstation system to enterprise-wide systems.

For further information contact your local IBM representative.

* * *

ManTech International has expanded its SQL Conversion Workbench, a utility for converting existing database applications to DB2. Enhancements allow conversions from Informix, Microsoft SQL Server, and Sybase.

SQL Conversion Workbench helps create an efficient database design and application code and takes advantage of the features and functions of DB2 databases. It converts both the application code and database, providing tools for refining the converted database, mapping the existing data structure to the new environment, resolving discrepancies between the database products, and converting the application code.

For further information contact:
ManTech Design and Development, 6810 Deer Path Road, Elkridge, MD 21075, USA.
Tel: (410) 579 8240.
URL http://www.mantech.com.

* * *

xephon