86

DB2 update

December 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: trevore@xephon.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/db2update.html; you will need the user-id shown on your address label.

Editor

Trevor Eddolls

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.

New DB2 back-up procedure

We have developed a new back-up procedure for our shop. The back-up procedure is a set of JCL, calling REXX programs. It consists of six jobs, described below, where ‘xxx’ is the project name. For example, for the atm procedure, ‘xxx’ is atm:

- TCRxxx01creates the report-only image copy statements for all the tablespaces in the given database. It then executes the report-only image copy statements with the given lowest and highest change limits, and writes this output into a dataset.
- TCGxxx01 creates the image copy statements and the datasets according to the results of the report-only image copy job. If, according to the change limit parameters, an incremental image copy is needed, then an incremental image copy statement is prepared. Otherwise, if a full image copy is needed, then a full image copy statement is prepared. For the tablespaces for which an incremental or a full image copy is taken, the Stop tablespace and Start tablespace access (UT) commands, and the Start Tablespace Access (RW) statements are prepared. You can also make this job take full image copies by giving some parameters, and deciding on the tape environment that will be used.
- TCPxxx01 executes the image copy JCL prepared in the step above. Firstly, it checks whether there is at least one dataset whose image copy is to be taken. If there is, it stops the tablespaces whose image copies will be taken, and then starts them for the utility only. It then takes the image copies. Afterwards, it starts them for RW access.
- TMDxxx01 creates and executes the modified JCL. Every day, modify jobs for a given age are executed. This step first creates the modify statements for all the tablespaces in the given database, and then executes them.
- TMGxxx01 creates the merge copy JCL for the tablespaces. This job uses a REXX program that compares the start RBAs of the incremental and full image copies. If the start_rba of the

incremental image copy is higher than the full copy, then it decides that a merge will be done for this tablespace. Otherwise, it does not take this tablespace into consideration.

- TMCxxx01 executes the merge copy JCL that was created with the TMGxxx01 job.

For each database, the same set of JCL must be executed with different database name parameters. However, there is no need to specify the names of the tablespaces in the database because it checks the database for all the tablespaces.

TCRATM01

TCRATM01 JCL should reside in DBAT.COPY.JCL.

```
//TCRATMØ1 JOB ,'TABLE UNLOAD',
//   MSGLEVEL=(1,1),MSGCLASS=X,CLASS=D,REGION=6M,TIME=144Ø
/**********************************************************
//*   This job creates the report-only image copy statements
//*   then the report-only image copy step is executed.
//*   The output of this job is used for creating the image
//*   copy JCL - if they should be taken as incremental or full
//**********************************************************
/*JOBPARM SYSAFF=BXØ8
//UNLOADTS EXEC PGM=IKJEFTØ1
//*********************************
//* This step takes an unload of the tablespace names for the
//* given database
/*********************************
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  DSN SYSTEM(DB2T)
    RUN  PROGRAM(DSNTIAUL) PLAN(DSNTIB51) PARMS('SQL') -
    LIB('DSNDBØT.RUNLIB.LOAD')
  END
/*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD DUMMY
//SYSPUNCH DD DUMMY
//SYSRECØØ DD DSN=DBAT.COPY.TSNAME(TDATMØØ1),DISP=SHR
//SYSIN    DD *
SELECT NAME FROM SYSIBM.SYSTABLESPACE WHERE DBNAME='TDATMØØ1'
;
//CRERPRT  EXEC PGM=IRXJCL,
//         PARM='COPY1 TDATMØØ1 1Ø 3Ø DB2T'
```

```
//*****************************************************
//* This step executes the Copy1 REXX program
//* Input:
//*   Inp: The tablespace names for the database that will be
//*        unloaded
//* Output:
//*    Stoprep: It creates the Stop Tablespace commands here
//*    Copyrep: It creates the report only image copy statements
//*    here
//*    Startrep: It creates the Start Tablespace Access(rw)
//*              statement here
//* Parameters:
//*    TDATMØØ1: The first parameter is the name of the database
//*              for which the back-up will be taken
//*    1Ø: This is the low limit of the changelimit value for the
//*        report-only image copy
//*    3Ø: This is the high limit of the changelimit value for the
//*        report-only image copy
//*    DB2T: This is the name of the DB2 subsystem in which the
//*          programs will be executed
//
**********************************************************************
//OUTDD    DD   SYSOUT=*
//SYSTSPRT DD   SYSOUT=*
//SYSEXEC  DD   DSN=DBAT.COPY.EXEC,DISP=SHR
//SYSOUT   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//INP      DD   DSN=DBAT.COPY.TSNAME(TDATMØØ1),DISP=SHR
//STOPREP  DD   DSN=DBAT.COPY.STOPREP(TDATMØØ1),DISP=SHR
//COPYREP  DD   DSN=DBAT.COPY.COPYREP(TDATMØØ1),DISP=SHR
//STARTREP DD   DSN=DBAT.COPY.STARTREP(TDATMØØ1),DISP=SHR
/*
//STOPTS    EXEC PGM=IKJEFTØ1
//
**********************************************************************
//* By using the output of Copy1 REXX program,
//* this step executes the Stop Tablespace commands
//* and the Start Tablespace Access(UT) commands
//* before executing the report-only image copy step
//
**********************************************************************
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD DSN=DBAT.COPY.STOPREP(TDATMØØ1),DISP=SHR
/*
//COPYRPRT EXEC DSNUPROC,SYSTEM=DB2T,UID=TDATMØØ1
//
**********************************************************************
//* This step executes the report-only image copy statements
//* and saves the output as a member in the DBAT.COPY.REPORT
//* dataset so that a later job can pass through and evaluate
//* the output of the report-only image copy step
//
*************************************************************************
//SYSPRINT DD DSN=DBAT.COPY.REPORT(TDATMØØ1),DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSCOPY  DD DUMMY
//UTPRINT  DD SYSOUT=*
//SYSIN    DD DSN=DBAT.COPY.COPYREP(TDATMØØ1),DISP=SHR
/*
//STARTTS   EXEC PGM=IKJEFTØ1
//
*************************************************************************
//* This step execute the Start Tablespace Access(rw) statements
//
*************************************************************************
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD DSN=DBAT.COPY.STARTREP(TDATMØØ1),DISP=SHR
/*
```

COPY1

The Copy1 REXX program should reside in DBAT.COPY.EXEC.

```
/* REXX */
/* Copy1 REXX Program*/
/* This program prepares the necessary stop tablespace            */
/* statements, report-only image copy statements, and start       */
/* tablespace statements                                          */
/* Input:                                                         */
/*   inp: The tablespace names in the given database that are     */
/*        unloaded with the tcratmØ1 job                          */
/* Output:                                                        */
/*   stoprep: The stop tablespace statements are written into     */
/*            that dataset                                        */
/*   copyrep: The report only image copy statements are written   */
/*   startrep: The start tablespace access(ut) statements are     */
/*             written                                            */
/* Parameters                                                     */
/*   dbname: The database name for which the image copies will be */
/*           taken                                                */
/*   per1: The lower changelimit ratio                            */
/*   per2: The upper changelimit ratio                            */
/*   db2id: The name of the DB2 subsystem                         */

 arg dbname per1 per2 db2id

 'execio * diskr inp (stem tinp. finis'
 'execio Ø diskw stoprep (finis'
 'execio Ø diskw copyrep  (finis'
 'execio Ø diskw startrep  (finis'
 /* For all the tablespaces given in the input file, prepare the    */
 /* stop tablespace commands                                         */
 k=1
 tstop.k='  DSN SYSTEM('||db2id||')'
 do i=1 to tinp.Ø
    k=k+1
    tstop.k='   -STOP  DATABASE('||dbname||') SPACENAM('||tinp.i||')'
    k=k+1
    tstop.k='   -START DATABASE('||dbname||') SPACENAM('||tinp.i
    tstop.k=tstop.k||')'||' ACCESS(UT)'
 end
 k=k+1
 tstop.k='   END'
 tstop.Ø=k
 /* For all the tablespaces given in the input file, prepare the */
 /* report-only image copy statements                            */
 k=Ø
 do i=1 to tinp.Ø
     k=k+1
     tcopy.k='    COPY TABLESPACE '||dbname||'.'||tinp.i
     k=k+1
     tcopy.k='    DSNUM ALL'
     k=k+1
     tcopy.k='    COPYDDN SYSCOPY'
     k=k+1
     tcopy.k='    CHANGELIMIT ('||per1||','||per2||')'
     k=k+1
     tcopy.k='    SHRLEVEL REFERENCE'
     k=k+1
     tcopy.k='    REPORTONLY'
 end
 tcopy.Ø=k
 /* For all the tablespaces given in the input file, prepare the */
 /* start tablespace access (rw) statements                      */
 k=1
 tstart.k='   DSN SYSTEM('||db2id||')'
 do i=1 to tinp.Ø
   k=k+1
   tstart.k='     -START DATABASE('||dbname||') SPACENAM('
   tstart.k=tsart.k||tinp.i||') ACCESS(RW)'
 end
 k=k+1
 tstart.k=' END'
 tstart.Ø=k
 /* Write the output files to disk                               */
    'execio * diskw startrep (stem tstart. '
    'execio * diskw copyrep (stem tcopy. '
    'execio * diskw stoprep (stem tstop. '
 exit
```

TCGATM01

TCGATM01 JCL should reside in DBAT.COPY.JCL.

```
//TCGATMØ1 JOB ,'Image copy create job',
//   MSGLEVEL=(1,1),MSGCLASS=X,CLASS=D,REGION=6M,TIME=144Ø
/*JOBPARM SYSAFF=BXØ8
//
************************************************************************
//* This job creates the image copy statements by using the
//* report-only image copy output
//
************************************************************************
//CRESTMT  EXEC PGM=IRXJCL,
//         PARM='COPY2 TDATMØØ1 DB2T NO G T ATM VTAPE'
//
************************************************************************
//* This step calls the copy2 REXX program and creates the image
//* copy statements
//* Input:
//*   Inp: This is the output of the report-only image copy job
//* Output:
//*   Startcop: The stop tablespace commands, and the start
//*             tablespace access (ut) commands for the tablespaces
//*             of which the back-up will be taken, are prepared
//*             here
//*   Copy: The image copy statements for the tablespaces are
//*         prepared here
//*   Copydb: The dataset descriptions for the image copies are
//*           prepared here
//*   Stopcop: The Start tablespace access (rw) statements are
//*            prepared here
//* Parameters:
//*   TDATMØØ1: The name of the database for which the image copy is
//*             taken
//*   DB2T: The name of the DB2 subsystem
//*   No: This parameter can either be YES or NO
//*         YES: If Yes is given, then regardless of the changelimit
//*              parameters, Full Yes image copy statements will be
//*              prepared
//*         NO: According to the changelimit parameters, either FULL NO
//*             or FULL YES statements will be prepared
//*   G: This parameter can be either G, H, A, or Y. It is used to
//*      determine the retention period of the image copy dataset.
//*         G: Daily image copy dataset. The retention period is 15
//*            days
//*         H: Weekly image copy dataset. The retention period is 1
//*            month
//*         A: Monthly image copy dataset. The retention period is 1
//*            year
//*         Y: Yearly image copy dataset. The retention period is 10
//*            years
//*   T: This parameter represents the environment of the subsystem. The
//*      vcats of the image copy datasets are determined by using this
//*      parameter
//*       D: Development
//*       T: Test
//*       A: User acceptance
//*       P: Production
//*   ATM: Name of the project
//*   VTAPE: The unit name of the image copy datasets
//
**************************************************************************
//OUTDD    DD   SYSOUT=*
//SYSTSPRT DD   SYSOUT=*
//SYSEXEC  DD   DSN=DBAT.COPY.EXEC,DISP=SHR
//UTPRINT  DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//INP      DD   DSN=DBAT.COPY.REPORT(TDATM001),DISP=SHR
//STARTCOP DD   DSN=DBAT.COPY.STOPCOP(TDATM001),DISP=SHR
//COPY     DD   DSN=DBAT.COPY.COPYDDN(TDATM001),DISP=SHR
//COPYDB   DD   DSN=DBAT.COPY.COPYSTMT(TDATM001),DISP=SHR
//STOPCOP  DD   DSN=DBAT.COPY.STARTCOP(TDATM001),DISP=SHR
/*
```

COPY2

The Copy2 REXX program should reside in DBAT.COPY.EXEC.

```
/* REXX */
/* Copy2 REXX Program*/
/* This program, by using the output of the report-only image copy  */
/* statements, decides which tablespaces will be image copied, and  */
/* if the image copies will be full image copies or incremental     */
/* image copies                                                     */
/* Input:                                                           */
/*    Inp: This is the output of the report-only image copy step    */
/* Output:                                                          */
/*    Startcopy: The stop tablespace and the start tablespace       */
/*               access (ut) statements for the tablespaces for     */
/*               which the image copies will be taken are prepared  */
/*    Copy: The copy statements for the tablespaces for which the   */
/*          image copies will be taken are prepared                 */
/*    Copydb: The datasets for the image copies are prepared        */
/*    Stopcop: The Start tablespace access (rw) statements are      */
/*             prepared                                             */
/* Parameters:                                                      */
/*    db2id: The name of the DB2 subsystem                          */
/*    fully: Fully decides if the image copies will be taken        */
/*           according to the changelimit parameters or not. Fully  */
/*           can be either Y or N                                   */
/*           Y: All the image copies will be taken FULL YES         */
/*              overwriting the changelimit parameters              */
/*           N: The image copies will be taken according to the     */
/*              changelimits                                        */
/*    period: The period of the image copy. It can be G, H, A, or Y */
/*            This parameter determines the retention period of     */
/*            the back-ups                                          */
/*            G: Daily image copy. Retention period is 15 days      */
/*            H: Weekly image copy. Retention period is 1 month     */
/*            A: Monthly image copy. Retention period is 1 year     */
/*            Y: Yearly image copy. Retention period is 10 years    */

 arg dbname db2id fully period level vcat unit
 'execio * diskr inp (stem tinp. finis'
 'execio 0 diskw startcop (finis'
 'execio 0 diskw copy     (finis'
 'execio 0 diskw copydb   (finis'
 'execio 0 diskw stopcop  (finis'
/* Select the retention periods for different kinds of image copies   */
/* If it is a daily image copy (G), then the retention period will    */
/* be 15 days. If it is a weekly image copy (H), then the retention   */
/* period will be 1 month. If it is a monthly image copy (A), then    */
/* the retention period will be 1 year. If it is a yearly image       */
/* copy (Y), then the retention period will be 10 years.              */
 select
   when period=G then
     retpd=15
   when period=H then
     retpd=30
   when period=A then
     retpd=365
   when period=Y then
     retpd=3650
   otherwise do
     period=G
     retpd=8
   end /* otherwise */
 end   /* select    */

/* In our shop, the vcats of the GDGs of the image copies start with  */
/* B (for back-up) + project level (ex: ATM)                          */
 vcat='B'||vcat||level
j=0
```

```
/* This loop will check the report-only image copy output.           */
/* It starts checking the report output from the 4th line            */
/* It looks for the messages dsnu443i and dsnu446i                   */
/* Then checks if the image copy for this tablespace must be taken.  */
/* If it will be taken, then full or incremental is checked          */
do i=4 to tinp.Ø-1
  check1=find(tinp.i,dsnu443i)
  check2=find(tinp.i,dsnu446i)
  if (check1 | check2) then do
    j=j+1
    k=i-2
    select
      when subword(tinp.i,5,1)=='INCREMENTAL' then do
        copytype.j=NO
        if fully==YES then
          copytype.j=YES
        tsname.j=strip(subword(tinp.k,2,1),t,' ')
      end  /* when */
      when subword(tinp.i,5,1)=='FULL' then do
        copytype.j=YES
        tsname.j=strip(subword(tinp.k,2,1),t,' ')
      end /* when */
      when subword(tinp.i,5,1)=='NO' then do
        if fully==YES then do
          copytype.j=YES
          tsname.j=strip(subword(tinp.k,2,1),t,' ')
        end /* if */
        else
          j=j-1
      end /* when */
      otherwise do
        copytype.j=YES
        tsname.j=substr(subword(tinp.k,6,1),1Ø)
      end /* otherwise */
    end /* select */
  end /* if */
end /* do */
tsname.Ø=j
copytype.Ø=j
if tsname.Ø==Ø then
  exit
/* Prepare the Stop tablespace and Start tablespace access (ut)  */
/* statements                                                    */
k=1
tstart.k='  DSN SYSTEM('||db2id||')'
do i=1 to tsname.Ø
   k=k+1
   tstart.k='   -STOP  DATABASE('||dbname||') SPACENAM('||tsname.i||')'
   k=k+1
   tstart.k='   -START DATABASE('||dbname||') SPACENAM('||tsname.i
   tstart.k=tstart.k||')'||' ACCESS(UT)'
end
k=k+1
tstart.k='   END'
tstart.Ø=k
/* Prepare the image copy dataset definitions  */
k=Ø
do i=1 to tsname.Ø
  k=k+1
  tcopy.k='//'||tsname.i||' DD DSN='||vcat||'.'||tsname.i||'.'
  if copytype.i='YES' then
    tcopy.k=tcopy.k||'GIM'||period||'(+Ø1),'
  else
    tcopy.k=tcopy.k||'GIMI'||'(+Ø1),'
  k=k+1
  tcopy.k='//     DISP=(NEW,CATLG,CATLG),FREE=CLOSE,'
  k=k+1
  tcopy.k='//     VOL=(,RETAIN,,15'
  if i==1 then
    tcopy.k=tcopy.k||'),'
  else do
    j=i-1
    tcopy.k=tcopy.k||',REF=*.'||tsname.j||'),'
  end /* else */
  k=k+1
  tcopy.k='//     UNIT='||unit||',LABEL=('||i||',SL,RETPD='
  tcopy.k=tcopy.k||retpd||'),'
  k=k+1
  tcopy.k='//     DCB=((BKP.DB2.GDCB),TRTCH=COMP,BUFNO=2Ø)'
end /* do */
tcopy.Ø=k
/* Prepare the image copy statements   */
k=Ø
do i=1 to tsname.Ø
    k=k+1
    tcopydb.k='    COPY TABLESPACE '||dbname||'.'||tsname.i
    k=k+1
    tcopydb.k='    DSNUM ALL'
    k=k+1
    tcopydb.k='    FULL '||copytype.i
    k=k+1
    tcopydb.k='    COPYDDN '||tsname.i
    k=k+1
    tcopydb.k='    SHRLEVEL REFERENCE'
end
copydb.Ø=k
/* Prepare the start tablespace access(rw) statements */
k=1
tstop.k='   DSN SYSTEM('||db2id||')'
do i=1 to tsname.Ø
   k=k+1
   tstop.k='   -START DATABASE('||dbname||') SPACENAM('||tsname.i
   tstop.k=tstop.k||')'||' ACCESS(RW)'
end
if tsname.Ø==Ø then do
  exit code(12)
end
k=k+1
tstop.k=' END'
tstop.Ø=k
/* If full copy for all the tablespaces should be taken, then the    */
/* report output is not important, FULL is written to the report     */
/* output as a sign. The next JCL will check that and if FULL is     */
/* written, then the JCL will not be executed*/
if fully==YES then do
  'execio Ø diskw startcop (finis'
  tout.1=full
  tout.Ø=1
  'execio * diskw inp (stem tout. '
end
   'execio * diskw startcop (stem tstart. '
   'execio * diskw copy (stem tcopy. '
   'execio * diskw copydb (stem tcopydb. '
   'execio * diskw stopcop (stem tstop. '
exit
```

TCPATM01

TCPATM01 JCL should reside in DBAT.COPY.JCL.

```
//TCPATMØ1 JOB ,'TABLE UNLOAD',
//   MSGLEVEL=(1,1),MSGCLASS=X,CLASS=D,REGION=6M,TIME=144Ø
/*JOBPARM SYSAFF=BXØ8
//
***********************************************************************
//* This program takes the image copies
//
***********************************************************************
//PROC JCLLIB ORDER=DBAT.COPY.COPYDDN
//
***********************************************************************
//* This step checks, by using the copy3 REXX program, if
//* any image copy will be taken or not
//* If no tablespace in the database should be image copied
//* then return code 1 is returned from the COPY3 REXX program
//* Input:
//*    INP: FULL was written into the report output by the Copy2
//*         program if there were no tablespaces to be image
//*         copied. If FULL is read, then the image copies will
//*         not be taken
//
```

```
*************************************************************************
//CHECK    EXEC PGM=IRXJCL,
//         PARM='COPY3'
//OUTDD    DD   SYSOUT=*
//SYSTSPRT DD   SYSOUT=*
//SYSEXEC  DD   DSN=DBAT.COPY.EXEC,DISP=SHR
//SYSOUT   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//INP      DD   DSN=DBAT.COPY.REPORT(TDATMØØ1),DISP=SHR
/*
//
*************************************************************************
//* This step stops and starts the tablespaces with access(ut)
//* only if there are some tablespaces whose image copy will
//* be taken
//
*************************************************************************
//STOPTS    EXEC PGM=IKJEFTØ1,COND=(1,EQ)
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD DSN=DBAT.COPY.STOPCOP(TDATMØØ1),DISP=SHR
/*
//
*************************************************************************
//* This step takes the image copies only if there are some
//* tablespaces whose image copy will be taken
//
*************************************************************************
//COPY EXEC DSNUPROC,SYSTEM=DB2T,UID=TDATMØØ1,COND=(1,EQ)
//SYSPRINT DD SYSOUT=*
// INCLUDE MEMBER=TDATMØØ1
//SYSIN    DD DSN=DBAT.COPY.COPYSTMT(TDATMØØ1),DISP=SHR
/*
//
*************************************************************************
//* This step starts the tablespaces with access (rw)
//* only if there are some tablespaces whose image copy will
//* be taken
//
*************************************************************************
//STARTTS   EXEC PGM=IKJEFTØ1,COND=(1,EQ)
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD DSN=DBAT.COPY.STARTCOP(TDATMØØ1),DISP=SHR
/*
```

COPY3

The Copy3 REXX program should reside in DBAT.COPY.EXEC.

```
/* REXX */
/* Copy3*/
/* This program checks whether there is any tablespace for which the  */
/* back-up will be taken. If no back-up is needed for this database   */
/* then 1 is returned                                                 */
/* Input:                                                             */
/*   Inp: This is the report output member. If there were no          */
/*        tablespaces whose image copy should be taken, The Copy2     */
/*        program writes FULL into that member. This program checks   */
/*        only if FULL is written there or not. If FULL is written,   */
/*        then 1 is returned as a return code.                        */
 'execio * diskr inp (stem tinp. finis'
 i=tinp.1
 if tinp.1==FULL then
   exit
 else do
   i=tinp.Ø
   parse var tinp.i word1 +71 code
   if code==1 then
     exit(1)
 end
```

TMDATM01

TMDATM01 JCL should reside in DBAT.COPY.JCL.

```
//TMDATMØ1 JOB ,'TABLE UNLOAD',
//   MSGLEVEL=(1,1),MSGCLASS=X,CLASS=D,REGION=6M,TIME=144Ø
/*JOBPARM SYSAFF=BXØ8
//
************************************************************************
//* This job creates the modify JCL and then executes the
//* modifications
//
************************************************************************
//MODIFYRP EXEC PGM=IRXJCL,
//         PARM='MODIFY TDATMØØ1 15'
//
************************************************************************
//* This step creates the modify JCL by executing the modify
//* REXX program
//* Input:
//*    Inp: The tablespace names in the database are taken as input
//*    Modify: The modify JCL is created here
//* Parameters:
//*    TDATMØØ1: The name of the database
//*    15: The age of modifies
//
************************************************************************
```

```
//OUTDD    DD   SYSOUT=*
//SYSTSPRT DD   SYSOUT=*
//SYSEXEC  DD   DSN=DBAT.COPY.EXEC,DISP=SHR
//SYSOUT   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//INP      DD   DSN=DBAT.COPY.TSNAME(TDATMØØ1),DISP=SHR
//MODIFY   DD   DSN=DBAT.COPY.MODIFY(TDATMØØ1),DISP=SHR
/*
//MODIFY   EXEC DSNUPROC,SYSTEM=DB2T,UID=TDATMØØ1
//
************************************************************************
//* This step executes the modify JCL created in the above step
//
************************************************************************
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//UTPRINT  DD SYSOUT=*
//SYSIN    DD DSN=DBAT.COPY.MODIFY(TDATMØØ1),DISP=SHR
/*
```

MODIFY

The Modify REXX should reside in DBAT.COPY.EXEC.

```
/* REXX */
/* Modify*/
/* This REXX program creates the modify JCL for all the tablespaces  */
/* in the given database                                             */
/* Input:                                                            */
/*    Inp: The input is the name of the tablespaces in the database  */
/* Output:                                                           */
/*    Modify: The modified JCL is written into that member           */
/* Parameters:                                                       */
/*    Dbname: The name of the database                               */
/*    Modify: The age of the modification                            */
 arg dbname age
 'execio * diskr inp (stem tinp. finis'
 'execio Ø diskw modify (finis'
/* For all the tablespaces in the database, the modify statements are*/
/* prepared
 k=Ø
 do i=1 to tinp.Ø
     k=k+1
     tmodify.k='    MODIFY RECOVERY TABLESPACE '||dbname||'.'||tinp.i
     k=k+1
     tmodify.k='    DSNUM ALL'
     k=k+1
     tmodify.k='    DELETE AGE '||age
 end
```

```
k=k+1
tmodify.Ø=k
   'execio * diskw modify (stem tmodify. '
exit
```

TMGATM01

TMGATM01 JCL resides in DBAT.COPY.JCL.

```
//TMGATMØ1 JOB ,'TABLE UNLOAD',
//   MSGLEVEL=(1,1),MSGCLASS=X,CLASS=D,REGION=6M,TIME=144Ø
/*JOBPARM SYSAFF=BXØ8
//
************************************************************************
//* This program prepares the merge JCL by using the Merge REXX program
//
************************************************************************
//UNLDINC  EXEC PGM=IKJEFTØ1
//
************************************************************************
//* This step takes the unload of all tablespace names and the maximum
//* start_rbas of the incremental image copies in the given database
//
************************************************************************
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  DSN SYSTEM(DB2T)
    RUN  PROGRAM(DSNTIAUL) PLAN(DSNTIB51) PARMS('SQL') -
    LIB('DSNDBØT.RUNLIB.LOAD')
  END
/*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD DUMMY
//SYSPUNCH DD DUMMY
//SYSRECØØ DD DSN=DBAT.COPY.INCCOPY(TDATMØØ1),DISP=SHR
//SYSIN    DD *
SELECT TSNAME,MAX(HEX(START_RBA))
FROM SYSIBM.SYSCOPY
WHERE ICTYPE='I'
AND DBNAME='TDATMØØ1'
GROUP BY TSNAME
;
//UNLDFULL EXEC PGM=IKJEFTØ1
//
************************************************************************
//* This step takes the unload of all tablespace names and the maximum
//* start_rbas of the full image copies in the given database
//
************************************************************************
```

```
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  DSN SYSTEM(DB2T)
    RUN  PROGRAM(DSNTIAUL) PLAN(DSNTIB51) PARMS('SQL') -
    LIB('DSNDBØT.RUNLIB.LOAD')
  END
/*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD DUMMY
//SYSPUNCH DD DUMMY
//SYSRECØØ DD DSN=DBAT.COPY.FULCOPY(TDATMØØ1),DISP=SHR
//SYSIN    DD *
SELECT TSNAME,MAX(HEX(START_RBA))
FROM SYSIBM.SYSCOPY
WHERE ICTYPE='F'
AND DBNAME='TDATMØØ1'
GROUP BY TSNAME
;
/*
//CREMERG  EXEC PGM=IRXJCL,
//         PARM='MERGE TDATMØØ1 ATM T 15 VTAPE'
//
************************************************************************
//* This step uses the Merge REXX program to prepare the merge
//* statements and the merge dataset names
//* Input:
//*    Inp1: The unload of the tablespace names with their maximum
//*          start_rbas for the incremental image copies
//*    Inp2: The unload of the tablespace names with their maximum
//*          start_rbas for the full image copies
//* Output:
//*    Copyddn: The dataset description for the merge copy JCL is
//*             prepared in that member
//*    Copystmt: The statements for the merge copy JCL is prepared in
//*              that member
//* Parameters:
//*    TdatmØØ1: Name of the database on which the merge will be done
//*    Atm: The name of the project
//*    T: The level of the project
//*       D for development, T for Test, A for user acceptance, and P
//*       for production
//*    15: The retention period of the merged image copies
//*    Vtape: The device type on which the merged image copies will be
//*           created
//
************************************************************************
//OUTDD    DD   SYSOUT=*
//SYSTSPRT DD   SYSOUT=*
//SYSEXEC  DD   DSN=DBAT.COPY.EXEC,DISP=SHR
```

```
//SYSOUT   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//INP1     DD   DSN=DBAT.COPY.INCCOPY(TDATMØØ1),DISP=SHR
//INP2     DD   DSN=DBAT.COPY.FULCOPY(TDATMØØ1),DISP=SHR
//COPYDDN  DD   DSN=DBAT.COPY.MERGDDN(TDATMØØ1),DISP=SHR
//COPYSTMT DD   DSN=DBAT.COPY.MERGSTMT(TDATMØØ1),DISP=SHR
/*
```

MERGE

The Merge REXX should reside in DBAT.COPY.EXEC.

```
/* REXX */
/* Merge*/
/* This program creates the merge copy JCL                          */
/* Input:                                                           */
/*    Inp1: The unload of all tablespace names and the maximum      */
/*          start_rbas of the incremental image copies in the       */
/*          given database                                          */
/*    Inp2: The unload of all tablespace names and the maximum      */
/*          start_rbas of the full image copies in the given        */
/*          database                                                */
/* Output:                                                          */
/*    Copyddn: The merge copy dataset definitions will be created  */
/*    Copystmt: The merge copy statements will be created here      */
/* Parameters:                                                      */
/*    Dbname: The name of the database                              */
/*    vcat: The project name                                        */
/*    level: The level of the project                               */
/*       D: Development                                             */
/*       T: Test                                                    */
/*       A: User acceptance                                         */
/*       P: Production                                              */
/*    retpd: The retention period of the merged image copies        */
/*    unit: The device which will be used for the merge copies      */
 arg dbname vcat level retpd unit
 'execio * diskr inp1 (stem tinp1. finis'
 'execio * diskr inp2 (stem tinp2. finis'
 'execio Ø diskw copyddn (finis'
 'execio Ø diskw copystmt (finis'
/* The vcat for the merge copies is created according to our        */
 vcat=B||vcat||level
 k=1
 l=1
 m=Ø

/* If there is nothing to merge then return code 5                   */
/* This return code is checked to submit the next job from opc        */
```

```
if tinp1.Ø==Ø then
   exit(5)
/* This loop checks for a tablespace, both the incremental image     */
/* copy and the full image copy. If the start_rba of the image       */
/* copy is more than the full image copy start_rba for a             */
/* tablespace, then the full and the incremental image copies of     */
/* that tablespace will be chosen to be merged. Otherwise, there     */
/* will be no merge for this tablespace                              */
/* Since the rbas are in hexadecimal notation, convert function      */
/* is used to convert the start rbas to decimal notation             */
do i=1 to tinp1.Ø
  parse var tinp1.i inctsname +8 incrba
  found=Ø
  do until found
    parse var tinp2.l fultsname +8 fulrba
      if inctsname==fultsname then do
        found=1
        incrba=convert(incrba)
        fulrba=convert(fulrba)
        if incrba>fulrba then do
          m=m+1
          tout.m=fultsname
        end
      end
    l=l+1
  end
end
tout.Ø=m
/* If no merge will be done, then return code 5 is given */
if m==Ø then
  exit(5)
else do
  do i=1 to tout.Ø
     end
end
/* Prepare the merge copy statements for the tablespaces whose full  */
/* copies will be merged with their incremental image copies to make */
/* a full image copy                                                 */
k=Ø
do i=1 to tout.Ø
    k=k+1
    tout.i=strip(tout.i)
    tcopystmt.k='    MERGECOPY TABLESPACE '||dbname||'.'||tout.i
    say tcopystmt.k
    k=k+1
    tcopystmt.k='    COPYDDN ('||tout.i||')'
    say tcopystmt.k
    k=k+1
    tcopystmt.k='    NEWCOPY YES '
    say tcopystmt.k
end

/* Prepare necessary dataset definitions for the tablespaces whose  */
/* full image copies will be merged with their incremental image    */
/* copies to make a full image copy                                  */
k=Ø
do i=1 to tout.Ø
    k=k+1
    tout.i=strip(tout.i)
    tcopyddn.k='//'||tout.i||' DD DSN='||vcat||'.'||tout.i||'.'
    tcopyddn.k=tcopyddn.k||'GIMG(+Ø1),'
    say tcopyddn.k
    k=k+1
    tcopyddn.k='//     DISP=(NEW,CATLG,CATLG),FREE=CLOSE,'
    say tcopyddn.k
    k=k+1
    tcopyddn.k='//     VOL=(,RETAIN,,15'
    if i==1 then
      tcopyddn.k=tcopyddn.k||'),'
    else do
      j=i-1
      tcopyddn.k=tcopyddn.k||',REF=*.'||tout.j||'),'
    end /* else */
    say tcopyddn.k
    k=k+1
    tcopyddn.k='//     UNIT='||unit||',LABEL=('||i||',SL,RETPD='
    tcopyddn.k=tcopyddn.k||retpd||'),'
    say tcopyddn.k
    k=k+1
    tcopyddn.k='//     DCB=((BKP.DB2.GDCB),TRTCH=COMP,BUFNO=2Ø)'
    say tcopyddn.k
end
  tcopyddn.Ø=k-1
  'execio * diskw copyddn (stem tcopyddn. '
  'execio * diskw copystmt (stem tcopystmt. '
 exit
Convert: procedure
/* This procedure converts the hexadecimal start_rba into decimal  */
/* notation                                                        */
/* Input:                                                          */
/*    rba: The start_rba in hexadecimal notation                   */
/* Output:                                                         */
/*    resultrba: The start_rba converted to decimal notation is    */
/*               returned                                          */
arg rba
do i=1 to 12
  rba.i=substr(rba,i,1)
  select
```

```
    when rba.i=A then
      rba.i=10
    when rba.i=B then
      rba.i=11
    when rba.i=C then
      rba.i=12
    when rba.i=D then
      rba.i=13
    when rba.i=E then
      rba.i=14
    when rba.i=F then
      rba.i=15
    otherwise nop
  end
end
j=0
resultrba=0
do i=12 to 1 by -1
  resultrba=rba.i*(16**j)+resultrba
  j=j+1
end
return resultrba
```

TMCATM01

TMCATM01 JCL should reside in DBAT.COPY.EXEC.

```
//TMCATM01 JOB ,'TABLE UNLOAD',
//   MSGLEVEL=(1,1),MSGCLASS=7,CLASS=D,REGION=6M,TIME=1440
/*JOBPARM SYSAFF=BX08
//
************************************************************************
//* This program is the merge copy JCL for all the tablespaces
//* in the database
//
************************************************************************
//PROC JCLLIB ORDER=DBAT.COPY.MERGDDN
//COPY EXEC DSNUPROC,SYSTEM=DB2T,UID=TDATM001
//SYSPRINT DD SYSOUT=*
// INCLUDE MEMBER=TDATM001
//SYSIN    DD DSN=DBAT.COPY.MERGSTMT(TDATM001),DISP=SHR
/*
```

Banu Bayraktar Ekiz
DB2 Systems Programmer
Bilpa A S (Turkey)

A relational schema for a data warehouse

Editor's note: this article provides specific recommendations for DB2; however, these techniques can be applied to any relational database subject to individual RDBMS rules and syntax.

A previous article, *Using a relational database for data warehouses*, *DB2 Update*, Issue 83, September 1999, defined what a data warehouse is and why the standard OLTP relational schema causes poor performance. It recommended a special relational schema known as a star or a variation of a star. This article takes a highly normalized OLTP schema and decomposes it to a melted snowflake!

OLTP LOGICAL RELATIONAL DATABASE

The following code contains the logical relational database that I shall refer to – the Logic University Logical Relational Database. It has been generalized for supertypes and subtypes to eliminate duplicate attributes caused by storing common subtype attributes redundantly instead of once.

```
AEIS(AEIS-SAPK, <role>)
  AEIS_ADDRESS_PHONE(AEIS-SAPK, DATESTAMP, UOM, phones, city, state,
  postal_code, country, street_address)
  AEIS_DEG_PREVIOUS(AEIS-SAPK, DEGREE, GRADUATION-DATE, INSTITUTION-ID)
  AEIS_NAME(AEIS-SAPK, DATESTAMP, surname, name_rest)
    ALUMNUS (separate ALUMNUS subtable not needed - no attribs apply)
    ALU_EMP_CONTRIBUTE(AEIS-SAPK, DATESTAMP, amount)
    ALUMNUS_EMPLOYER(AEIS-SAPK, <OD>, AEIS_SAPK, employer)
    ALUMNUS_STUDENT(AEIS-SAPK, SCHOOL-ID + student-od)
  EMPLOYER(AEIS-SAPK, employer-id)
  INSTRUCTOR(AEIS-SAPK, SCHOOL_ID + instructor_od, rank,
   parkinglot_space, employment_date, tenure_date)
INS_COU_HISTORY(AEIS-SAPK, SEMESTER-SAPK, COURSE-SAPK, <rating>)
INS_STU_SCHEDULE(AEIS-SAPK, WEEKDAY & TIME-START, COURSE_SAPK,
CLASSROOM_SAPK)
INSTUCTOR_COURSE(AEIS-SAPK, COURSE-SAPK, teach_backup)
  STUDENT(AEIS-SAPK, AEIS_SAPK_advisor, status, sat_score,
    grad_date_expected, enrollment_date, postmark_date)
    STU_COU_HISTORY(AEIS-SAPK, COURSE-SAPK, SEMESTER-SAPK, grade,
    hours_taken, AEIS_SAPK_ins)
    STUDENT_CURRICULUM(AEIS-SAPK, CURRICULUM-SAPK, major_minor)
BOOK(BOOK-SAPK, publisher_id + book_od, dewey, title)
  BOOK_AUTHOR(BOOK-SAPK, AUTHOR-NAME)
  BOOK_COURSE_USING(BOOK-SAPK, COUNTRY-ID, COURSE-SAPK)
```

```
CLASSROOM(CLASSROOM-SAPK, floor_id + room_od, capacity)
  CLA_EQU(CLASSROOM-SAPK, <OD>, EQUIPMENT-ID)
COURSE(COURSE-SAPK, SCHOOL_ID + sequence_od + stream_od, course-name,
min_hours, max_hours, min_classize, max_classize, hour_duration,
<rating>)
  COURSE_EQUIPMENT(COURSE-SAPK, EQUIPMENT-ID)
CURRICULUM(CURRICULUM-SAPK, curriculum_name + SCHOOL_ID, major_hours,
minor_hours, highest_stream)
EQUIPMENT(EQUIPMENT-ID, name)
INSTITUTION(INSTITUTION-ID, name)
SCHOOL(SCHOOL-ID, school_name)
SEMESTER(SEMESTER-SAPK, season_name + mcdy, date_start, date_finish)
```

A person can play four roles (subtypes) at Logic University (LU): Alumnus, Employer, Instructor, and Student (AEIS). A graduate student who instructs plays the three roles of alumnus, student, and instructor. Separate tables would store attributes such as name, address, phone, etc multiple times instead of once.

For example, getting graduate student details would require a join of AEIS for role:

- AEIS_ADDRESS_PHONE for address and phone.
- AEIS_DEG_PREVIOUS for degree, graduation_date, and (granting) institution_id.
- AEIS_NAME for surname and name_rest.

It could require over twenty joins plus subqueries for multivalue attributes. DBAs refer to this as ‘the query from hell’ because it requires immense computer resources to execute. The problem is that the database and its resultant schema has been designed for OLTP and not data warehouses.

A data warehouse has to serve defined information needs. For example, LU is American and has applied for a Baldrige National Quality Award (awarded to institutes that achieve educational excellence). A major evaluation criterion is ‘Student and Stakeholder’ focus. This includes:

- How the school learns from its former, current, and future students.
- How the school monitors student utilization of offerings, facilities, and services.

- Demographic data and trends that may bear upon enrolments and needs.

There are many other requirements, but we will focus on these three.

The first step in building a data warehouse is determining where the required data is. Students provide instructor ratings (INSTRUCTOR) and course ratings (COURSE) that provide learning. ALUMNUS instructor and course ratings can be tracked by including their STUDENT recommendations.

This raises a common data warehouse question: how much history? LU's first graduating class was 1907 and it still has a few living alumni from that era. LU started its rating system in 1990. The logical choice is 1990 because there is no rating data before. STUDENT contains sat_score (a scholastic aptitude test to assess knowledge) that could be used to correlate their aptitude scores to their grade point average. The LU database does not contain any data on the curriculum wanted by future students requesting admission, although a possible result of Baldrige would be the inclusion of future curriculum data in the database.

Monitoring data can be provided by STU_COU_HISTORY, which holds data for all courses that all students have taken. History should probably be limited to 1990 to maintain conformity with the alumnus data. INS_STU_SCHEDULE contains foreign keys of COURSE_SAPK and CLASSROOM_SAPK that can provide book, classroom, equipment, and instructor utilization by correlating data in BOOK_COURSE_USING, CLASSROOM, CLA_EQUIPMENT, COURSE_EQUIPMENT, and EQUIPMENT.

Demographic data can be provided by AEIS_ADDRESS_PHONE.

It should be evident that using the OLTP schema will require many joins. The data warehouse data must be collected or amalgamated into a schema that requires fewer joins. It is possible to place all data into a single table, thereby eliminating joins – but the DASD storage cost is usually exorbitant. Placing data into tables can identify problems.

```
DATA_WAREHOUSE(AEIS-SAPK, datestamp, uom, address, phone, datestamp,
uom, address, phone, datestamp...
```

LU must keep track of all addresses and phone numbers that a person has ever had, including data of change (datestamp) and whether it is home, college, vacation, etc (unit of measure). This creates the old problem of guessing how many iterations are required for address and phone. For example, assume ten is chosen; this gives rise to three problems:

- It does not matter what ‘n’ is set to because there will always be ‘n+1’.
- Any person not using n will require nulls in the unused iterations, wasting DASD.
- The quantity of nulls will require complex processing.

Continuing on, degree, graduation_date, institution_id is another instance of iteration; datestamp, surname requires iteration; and so forth. One table does not work. A star schema requires the separation of data into facts (what is to be measured) and dimensions (how to measure). The ‘learning’ facts are:

- Rating for instructor
- Rating for course
- sat_score.

Standard dimensions include time and location. The instructor rating has a date (SEMESTER_SAPK) but course rating does not because it is a running average. Student sat_score has a date (enrolment_date). There is a single location that is LU itself, so a separate dimension table is not required for location. LU does require entity dimkeys.

```
LEARN_FACT(date_dimkey, instructor_dimkey, instructor_rating,
course_dimkey, course_rating, student_dimkey, sat_score)
DATE_DIM(date_dimkey, year, semester)
```

date_dimkey is the ‘foreign key’ linking the tables. Assuming that the rating system started in the fall semester of 1990 then the values are 1, 1990, fall. The next values are 2, 1991, spring. The dimkey is an odometer counting instances. LU would have started its 28th iteration this fall (28, 1999, fall). The other dim tables are:

```
INSTRUCTOR_DIM(instructor_dimkey, aeis_sapk)
COURSE_DIM(course_dimkey, course_sapk)
STUDENT_DIM(student_dimkey, aeis_sapk)
```

The ‘demographic’ fact is address. LU must decide whose address, which address, how much history. They decide that only student home and college addresses are necessary while attending LU. Address components required are city, state, postal code, and country. They will start with fall 1990 for trend analysis.

```
DEMO_FACT(date_dimkey, student_dimkey, city_home, city_college, state ,
country, postal_code)
```

No new dim tables are required because they exist (DATE_DIM and STUDENT_DIM). The tables to date are:

```
LEARN_FACT(date_dimkey, instructor_dimkey, instructor_rating,
course_dimkey, course_rating, student_dimkey, sat_score)
DEMO_FACT(date_dimkey, student_dimkey, city_home, city_college, state,
country, postal_code)

DATE_DIM(date_dimkey, year, semester)
INSTRUCTOR_DIM(instructor_dimkey, aeis_sapk)
COURSE_DIM(course_dimkey, course_sapk)
STUDENT_DIM(student_dimkey, aeis_sapk)
```

This is a star schema, but it is also a relational schema. It is vastly different from its OLTP schema, with the net result being fewer joins and faster response time for knowledge workers using the data warehouse. The star does require joins between the fact table and the appropriate dim tables. An objective of a star and its variations is to further reduce joins. This often requires in-depth knowledge of the data. LU dates are semester-based, not requiring a dim table. Subtyping has provided subtype sapks, making those dim tables redundant. The new tables are:

```
LEARN_FACT(semester, aeis_instructor, instructor_rating, course_sapk,
course_rating, aeis_student, sat_score)
DEMO_FACT(semester, aeis_student, city_home, city_college, state,
country, postal_code)
```

This combination is the star variation of a melted snowflake. Individual names are not required yet, but probably will be as more Baldrige criteria data is stored in the data warehouse. Instructor name can be stored in LEARN_FACT and student name in DEMO_FACT. If alumnus name is needed then a new table is created:

```
ALUM_FACT(aeis_alumnus, name)
```

Physical considerations include using simple tablespace with clustering index and PCTFREE and FREEPAGE set to 0. The following construct will be used for discussion:

```
FACT(dimkey, fact1, fact2, fact3,...)
DIM(dimkey, dim1, dim2, dim3,...)
```

Simple tablespace was the only tablespace until Version 1.4, but is rarely used today because segmented tablespaces offer significant advantages. Almost the only reason today to use simple is to mix rows of multiple tables, such as FACT and DIM, in a single table. The procedure is:

- Create a tablespace as a simple tablespace.
- Create FACT and DIM, assigning them to the newly created simple tablespace.
- Create dimkey as the clustering index for both tables.
- Sort both tables by the clustering dimkey index.
- Mix FACT and DIM rows by inserting FACT then DIM until done.

This physical construct allows data warehouse software to do a join of FACT and DIM without requiring a physical I/O, thereby improving response time. A typical star schema will have multiple fact tables, each with multiple dim tables. Defining a mixing strategy for star schemas can be complex, but the gain in performance is worth the effort.

I suspect that some readers thought that 0 for PCTFREE and FREEPAGE was a typo but it is not. The data warehouse is read only, with new data being appended:

- INSERT is never done into existing rows.
- UPDATE is never done.
- DELETE is only done for data that is being offloaded because it is too old, or for other reasons.

The IUD usage above does not require free space and so 0 is the correct percentage. Finally, star tables do not require a primary key because there is no RI. dimkey or an appropriate combination should be declared as UNIQUE.

Eric Garrigue Vesely
Principal
Analyst Workbench Consulting (Malaysia)

Quick table information – part 3

This month we conclude the code for the procedure that gives quick DB2 table information, column, index, and referential integrity information, and generates a report.

PTINF03

```
* PROCESS GS,OFFSET,OPT(TIME);
 PTINF03:PROC(PARMS)OPTIONS(MAIN) REORDER;
 /*************************************************************/
 /* DESCRIPTION: PL/I PROGRAM - INDEX DETAIL                   */
 /*************************************************************/
  DCL PARMS CHAR(1ØØ) VAR;
  DCL SYSPRINT    FILE STREAM OUTPUT;
  DCL 1 WORKST,
      2 CRE         CHAR(8)  VAR,
      2 TAB         CHAR(18) VAR;
  /************************************************************/
  /* DCLGEN TABLE: SYSIBM.SYSINDEXES                          */
  /************************************************************/
  DCL 1 DCL1,
      5 NAME             CHAR(18) VAR,
      5 CREATOR          CHAR(8),
      5 TBNAME           CHAR(18) VAR,
      5 TBCREATOR        CHAR(8),
      5 UNIQUERULE       CHAR(1),
      5 COLCOUNT         BIN FIXED(15),
      5 CLUSTERING       CHAR(1),
      5 CLUSTERED        CHAR(1),
      5 DBID             BIN FIXED(15),
      5 OBID             BIN FIXED(15),
      5 ISOBID           BIN FIXED(15),
      5 DBNAME           CHAR(8),
      5 INDEXSPACE       CHAR(8),
      5 FIRSTKEYCARD     BIN FIXED(31),
      5 FULLKEYCARD      BIN FIXED(31),
      5 NLEAF            BIN FIXED(31),
      5 NLEVELS          BIN FIXED(15),
      5 BPOOL            CHAR(8),
      5 PGSIZE           BIN FIXED(15),
      5 ERASERULE        CHAR(1),
      5 DSETPASS         CHAR(8),
      5 CLOSERULE        CHAR(1),
      5 SPACE            BIN FIXED(31),
      5 IBMREQD          CHAR(1),
      5 CLUSTERRATIO     BIN FIXED(15),
      5 CREATEDBY        CHAR(8),
      5 IOFACTOR         BIN FIXED(15),
      5 PREFETCHFACTOR   BIN FIXED(15),
      5 STATSTIME        CHAR(26),
      5 INDEXTYPE        CHAR(1);
  DCL 1 DCL2,
      5 COLNAME     CHAR(18) VAR,
      5 COLSEQ      BIN FIXED(15),
      5 ORDERING    CHAR(1),
      5 COLCARD     BIN FIXED(31);

  DCL (SUBSTR,DATE,TIME,NULL,ADDR,LENGTH,INDEX) BUILTIN;

  CRE=SUBSTR(PARMS,1,8);
  TAB=SUBSTR(PARMS,9,18);

  PUT SKIP LIST('INDEX INFORMATION');
  EXEC SQL INCLUDE SQLCA;

  /* SELECTION RESULTS                              */
  EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR SELECT
    NAME, CREATOR, UNIQUERULE, CLUSTERING, CLUSTERED,
    CLUSTERRATIO, BPOOL, INDEXTYPE
  FROM SYSIBM.SYSINDEXES
  WHERE TBCREATOR = :CRE
    AND TBNAME    = :TAB
  ORDER BY CREATOR,NAME
  FOR FETCH ONLY;

  EXEC SQL OPEN C1;

  EXEC SQL FETCH C1 INTO
    :NAME, :CREATOR, :UNIQUERULE, :CLUSTERING, :CLUSTERED,
    :CLUSTERRATIO, :BPOOL, :INDEXTYPE;
  DO WHILE (SQLCODE=Ø);
     PUT SKIP LIST ('I '||NAME||' '||CREATOR||' '||UNIQUERULE||' '||
         CLUSTERING||' '||CLUSTERED||' '||CLUSTERRATIO||' '||BPOOL||
         ' '||INDEXTYPE);
     CALL KEYS;
     EXEC SQL FETCH C1 INTO
       :NAME, :CREATOR, :UNIQUERULE, :CLUSTERING, :CLUSTERED,
       :CLUSTERRATIO, :BPOOL, :INDEXTYPE;
  END;
  KEYS:PROC;
     EXEC SQL DECLARE C2 CURSOR WITH HOLD FOR SELECT
       COLNAME,ORDERING,COLCARD,COLSEQ
     FROM SYSIBM.SYSKEYS,
          SYSIBM.SYSCOLUMNS
     WHERE TBCREATOR = :CRE
       AND TBNAME    = :TAB
```

```
        AND COLNAME   = NAME
        AND IXCREATOR = :CREATOR
        AND IXNAME    = :NAME
      ORDER BY COLSEQ
      FOR FETCH ONLY;
      EXEC SQL OPEN C2;
      EXEC SQL FETCH C2 INTO :COLNAME, :ORDERING, :COLCARD, :COLSEQ;
      DO WHILE (SQLCODE=Ø);
         PUT SKIP LIST ('K '||COLNAME||' '||ORDERING||' '||COLCARD);
         EXEC SQL FETCH C2 INTO :COLNAME, :ORDERING, :COLCARD, :COLSEQ;
      END;
      EXEC SQL CLOSE C2;
   END KEYS;
   EXEC SQL CLOSE C1;
  END PTINFO3;
```

PTINF04

```
* PROCESS GS,OFFSET,OPT(TIME);
 PTINFO4:PROC(PARMS)OPTIONS(MAIN) REORDER;
 /*************************************************************/
 /* DESCRIPTION: PL/I PROGRAM - REFERENTIAL INTEGRITY          */
 /*************************************************************/
  DCL PARMS CHAR(1ØØ) VAR;
  DCL SYSPRINT    FILE STREAM OUTPUT;
  DCL 1 WORKST,
      2 CRE         CHAR(8)  VAR,
      2 TAB         CHAR(18) VAR;
 /***********************************************************/
 /* DCLGEN TABLE: SYSIBM.SYSRELS                             */
 /***********************************************************/
 DCL 1 DCL1,
     5 CREATOR        CHAR(8),
     5 TBNAME         CHAR(18),
     5 RELNAME        CHAR(8),
     5 REFTBNAME      CHAR(18),
     5 REFTBCREATOR   CHAR(8),
     5 DELETERULE     CHAR(1),
     5 RSHIP          CHAR(1);

  DCL (SUBSTR,DATE,TIME,NULL,ADDR,LENGTH,INDEX) BUILTIN;

  CRE=SUBSTR(PARMS,1,8);
  TAB=SUBSTR(PARMS,9,18);

  PUT SKIP LIST('REFERENTIAL INTEGRITY');
  EXEC SQL INCLUDE SQLCA;

  /* SELECTION RESULTS                              */
  EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR SELECT
  '1',REFTBCREATOR,REFTBNAME,RELNAME,DELETERULE
  FROM SYSIBM.SYSRELS
  WHERE CREATOR = :CRE
    AND TBNAME = :TAB
  UNION SELECT
  '2',CREATOR,TBNAME,RELNAME,DELETERULE
  FROM SYSIBM.SYSRELS
  WHERE REFTBCREATOR = :CRE
    AND REFTBNAME = :TAB
  ORDER BY 1,2,3
  FOR FETCH ONLY;
  EXEC SQL OPEN C1;
  EXEC SQL FETCH C1 INTO
    :RSHIP, :CREATOR, :TBNAME, :RELNAME, :DELETERULE;
  DO WHILE (SQLCODE=Ø);
     IF RSHIP='1'
     THEN RSHIP='P';
     ELSE RSHIP='C';
     PUT SKIP LIST (RSHIP||' '||CREATOR||' '||SUBSTR(TBNAME,1,18)
                    ||' '||RELNAME||' '||DELETERULE);
     EXEC SQL FETCH C1 INTO
       :RSHIP, :CREATOR, :TBNAME, :RELNAME, :DELETERULE;
  END;
  EXEC SQL CLOSE C1;
 END PTINFO4;
```

PTINF05

```
* PROCESS GS,OFFSET,OPT(TIME);
 PTINFO5:PROC(PARMS)OPTIONS(MAIN) REORDER;
 /******************************************************************/
 /* DESCRIPTION: PL/I PROGRAM - TABLE DOCUMENT                     */
 /******************************************************************/
  DCL PARMS CHAR(1ØØ) VAR;
  DCL SYSPRINT    FILE STREAM OUTPUT;
  DCL MCARD         PIC'–.–.–9';
  DCL FIELD         PIC'ZZZ9';
  DCL POLJE         PIC'ZZ9';
  /****************************************************************/
  /* DCLGEN TABLE: SYSIBM.SYSTABLES                               */
  /****************************************************************/
  DCL 1 DCLT,
      5 NAME          CHAR(18),
      5 CREATOR       CHAR(8),
      5 DBNAME        CHAR(8),
      5 TSNAME        CHAR(8),
      5 DBID          BIN FIXED(15),
      5 OBID          BIN FIXED(15),
      5 COLCOUNT      BIN FIXED(15),
      5 CARD          BIN FIXED(31),
      5 REMARKS       CHAR(125) VAR,
      5 LABEL         CHAR(3Ø) VAR,
      5 PARENTS       BIN FIXED(15),
      5 CHILDREN      BIN FIXED(15),
      5 RECLENGTH     BIN FIXED(15),
      5 CREATEDTS     CHAR(1Ø),
      5 ALTEREDTS     CHAR(1Ø),
      5 PCTROWCOMP    BIN FIXED(15),
      5 STATSTIME     CHAR(26);
  DCL 1 WORKST,
      2 CRE         CHAR(8)  VAR,
      2 TAB         CHAR(18) VAR;

  DCL (SUBSTR,DATE,TIME,NULL,ADDR,LENGTH,INDEX) BUILTIN;

  PUT SKIP LIST (' SHORT TABLE INFORMATION');
  EXEC SQL INCLUDE SQLCA;
  CRE=SUBSTR(PARMS,1,8);
  TAB=SUBSTR(PARMS,9,18);

  /* SELECTION RESULTS                              */
  EXEC SQL SELECT
    NAME, CREATOR, DBNAME, TSNAME, DBID, OBID, COLCOUNT, CARD
  , SUBSTR(REMARKS,1,1ØØ), PARENTS, CHILDREN, RECLENGTH, LABEL
  , DATE(CREATEDTS), DATE(ALTEREDTS), DATE(STATSTIME)
  INTO
    :NAME,:CREATOR,:DBNAME,:TSNAME,:DBID,:OBID,:COLCOUNT,
    :CARD, :REMARKS, :PARENTS, :CHILDREN, :RECLENGTH, :LABEL,
    :CREATEDTS, :ALTEREDTS, :STATSTIME
  FROM SYSIBM.SYSTABLES
  WHERE CREATOR = :CRE
    AND NAME    = :TAB;
  MCARD=CARD;
  PUT SKIP LIST (' '||(79)'-');
  PUT SKIP LIST
  (' TABLE     : '||NAME||'    NUMBER OF ROWS :'||MCARD);
  PUT SKIP LIST (' CREATOR   : '||CREATOR);
  PUT SKIP LIST (' REMARKS   : '||REMARKS);
  PUT SKIP LIST (' LABEL     : '||LABEL);
  PUT SKIP LIST (' '||(79)'-');
  PUT SKIP LIST
  (' TSNAME    : '||TSNAME||'              CREATEDTS : '||CREATEDTS);
  PUT SKIP LIST
  (' DBNAME    : '||DBNAME||'              ALTEREDTS : '||ALTEREDTS);
  FIELD=RECLENGTH;
  PUT SKIP LIST
  (' RECLENGTH :'||FIELD||'                   STATSTIME : '||STATSTIME);
  FIELD=PARENTS;
  PUT SKIP LIST (' PARENTS   :'||FIELD);
  FIELD=CHILDREN;
  PUT SKIP LIST (' CHILDREN  :'||FIELD);
```

```
 FIELD=DBID;
 PUT SKIP LIST (' DBID      :'||FIELD);
 FIELD=OBID;
 PUT SKIP LIST (' OBID      :'||FIELD);
 PUT SKIP LIST (' ');
 PUT SKIP LIST (' '||(79)'-');
 PUT SKIP LIST (' COLUMNS INFORMATION');
 PUT SKIP LIST (' '||(79)'-');
/****************************************************************/
/* DCLGEN TABLE: SYSIBM.SYSCOLUMNS                              */
/****************************************************************/
DCL LENGP            PIC'ZZZZZ9';
DCL SCALP            PIC'ZZZZ9';
DCL COLCP            PIC'ZZZZZZZZ9';
DCL 1 DCLC,
    5 NAMEC          CHAR(18),
    5 COLTYPE        CHAR(8),
    5 LENG           BIN FIXED(15),
    5 SCALE          BIN FIXED(15),
    5 COLNO          BIN FIXED(15),
    5 NULLS          CHAR(1),
    5 COLCARD        BIN FIXED(31),
    5 REMARKSC       CHAR(125) VAR;

 EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR SELECT
   NAME, COLTYPE, LENGTH, SCALE, NULLS,
   COLCARD, SUBSTR(REMARKS,1,35) ,COLNO
 FROM SYSIBM.SYSCOLUMNS
 WHERE TBCREATOR = :CRE
   AND TBNAME    = :TAB
 ORDER BY COLNO
 FOR FETCH ONLY;
 EXEC SQL OPEN C1;
 EXEC SQL FETCH C1 INTO
   :NAMEC, :COLTYPE, :LENG, :SCALE, :NULLS,
   :COLCARD, :REMARKSC, :COLNO;

  PUT SKIP LIST
 (' COLNAME            COLTYPE  LENGTH SCALE   COLCARD NULLS REMARKS');
  PUT SKIP LIST
 (' --------- ---- --- -- ---- --'||
  ' -----------');
 DO WHILE (SQLCODE=Ø);
    LENGP=LENG;
    SCALP=SCALE;
    COLCP=COLCARD;
    PUT SKIP LIST (' '||NAMEC||' '||COLTYPE||' '||LENGP||' '||
        SCALP||' '||COLCP||'     '||NULLS||' '||REMARKSC);
    EXEC SQL FETCH C1 INTO
      :NAMEC, :COLTYPE, :LENG, :SCALE, :NULLS,
      :COLCARD, :REMARKSC, :COLNO;
```

```
END;
EXEC SQL CLOSE C1;
PUT SKIP LIST (' ');
PUT SKIP LIST (' '||(79)'-');
PUT SKIP LIST (' INDEX INFORMATION');
PUT SKIP LIST (' '||(79)'-');
DCL 1 DCL1,
    5 NAMEI            CHAR(18),
    5 CREATORI         CHAR(8),
    5 UNIQUERULE       CHAR(1),
    5 CLUSTERING       CHAR(1),
    5 CLUSTERED        CHAR(1),
    5 DBIDI            BIN FIXED(15),
    5 OBIDI            BIN FIXED(15),
    5 BPOOL            CHAR(8),
    5 CLUSTERRATIO     BIN FIXED(15),
    5 INDEXTYPE        CHAR(1);
DCL 1 DCL2,
    5 COLNAME     CHAR(18),
    5 COLSEQ      BIN FIXED(15),
    5 ORDERING    CHAR(4),
    5 COLCARD1    BIN FIXED(31);
EXEC SQL DECLARE C2 CURSOR WITH HOLD FOR SELECT
  NAME, CREATOR, UNIQUERULE, CLUSTERING, CLUSTERED,
  DBID, OBID, CLUSTERRATIO, BPOOL, INDEXTYPE
FROM SYSIBM.SYSINDEXES
WHERE TBCREATOR = :CRE
  AND TBNAME    = :TAB
ORDER BY CREATOR,NAME
FOR FETCH ONLY;
EXEC SQL OPEN C2;
EXEC SQL FETCH C2 INTO
  :NAMEI, :CREATORI, :UNIQUERULE, :CLUSTERING, :CLUSTERED,
  :DBIDI, :OBIDI, :CLUSTERRATIO, :BPOOL, :INDEXTYPE;
DO WHILE (SQLCODE=Ø);
   IF INDEXTYPE=' ' THEN INDEXTYPE='-';
   PUT SKIP LIST
             (' INDEX : '||NAMEI||'        CREATOR    : '||CREATORI);
   PUT SKIP LIST ('         UNIQUERULE : '||UNIQUERULE||(11)' '||
                  ' CLUSTERING : '||CLUSTERING);
   FIELD=DBIDI;
   PUT SKIP LIST ('         DBID       :'||FIELD||(9)' '||
                  ' CLUSTERED  : '||CLUSTERED);
   FIELD=OBIDI;
   POLJE=CLUSTERRATIO;
   PUT SKIP LIST ('         OBID       :'||FIELD||(9)' '||
                  ' RATIO      :'||POLJE);
   PUT SKIP LIST ('         BPOOL      : '||BPOOL||(4)' '||
                  ' INDEXTYPE  : '||INDEXTYPE);
   CALL KEYS;
   EXEC SQL FETCH C2 INTO
     :NAMEI, :CREATORI, :UNIQUERULE, :CLUSTERING, :CLUSTERED,
     :DBIDI, :OBIDI, :CLUSTERRATIO, :BPOOL, :INDEXTYPE;
END;
KEYS:PROC;
   EXEC SQL DECLARE C3 CURSOR WITH HOLD FOR SELECT
     COLNAME,ORDERING,COLCARD,COLSEQ
   FROM SYSIBM.SYSKEYS,
        SYSIBM.SYSCOLUMNS
   WHERE TBCREATOR = :CRE
     AND TBNAME    = :TAB
     AND COLNAME   = NAME
     AND IXCREATOR = :CREATORI
     AND IXNAME    = :NAMEI
   ORDER BY COLSEQ
   FOR FETCH ONLY;

   EXEC SQL OPEN C3;

   PUT SKIP LIST
   ((9)' '||'COLNAME              ORDERING     COLCARD');
   PUT SKIP LIST
   ((9)' '||'---------    -----    -----');
   EXEC SQL FETCH C3 INTO :COLNAME, :ORDERING, :COLCARD1, :COLSEQ;
   DO WHILE (SQLCODE=Ø);
      IF ORDERING='A'
      THEN ORDERING='ASC';
      ELSE ORDERING='DESC';
      COLCP=COLCARD1;
      PUT SKIP LIST
      ((9)' '||COLNAME||'   '||ORDERING||'       '||COLCP);
      EXEC SQL FETCH C3 INTO :COLNAME, :ORDERING, :COLCARD1, :COLSEQ;
   END;
   PUT SKIP LIST(' ');
   EXEC SQL CLOSE C3;
END KEYS;
EXEC SQL CLOSE C2;
/*****************************************************************/
/* DCLGEN TABLE: SYSIBM.SYSRELS                                  */
/*****************************************************************/
DCL 1 DCLRI,
    5 CREATORR       CHAR(8),
    5 TBNAMER        CHAR(18),
    5 RELNAME        CHAR(8),
    5 DELETERULE     CHAR(1),
    5 RSHIP          CHAR(1);
DCL RNAME            CHAR(17);
DCL DNAME            CHAR(8);
DCL (I1,I2)          BIN FIXED(15);
PUT SKIP LIST (' '||(79)'-');
```

```
  PUT SKIP LIST (' REFERENTIAL INTEGRITY');
  PUT SKIP LIST (' '||(79)'-');
  EXEC SQL DECLARE C4 CURSOR WITH HOLD FOR SELECT
  '1',REFTBCREATOR,REFTBNAME,RELNAME,DELETERULE
  FROM SYSIBM.SYSRELS
  WHERE CREATOR = :CRE
    AND TBNAME  = :TAB
  UNION SELECT
  '2',CREATOR,TBNAME,RELNAME,DELETERULE
  FROM SYSIBM.SYSRELS
  WHERE REFTBCREATOR = :CRE
    AND REFTBNAME = :TAB
  ORDER BY 1,2,3
  FOR FETCH ONLY;
  EXEC SQL OPEN C4;

  EXEC SQL FETCH C4 INTO
    :RSHIP, :CREATORR, :TBNAMER, :RELNAME, :DELETERULE;
  I1,I2=Ø;
  PUT SKIP LIST
  ((18)' '||'CREATOR  TABLE NAME         RELNAME  DELETERULE ');
  PUT SKIP LIST
  ((18)' '||'——— ————————— ——— ———— ');
  DO WHILE (SQLCODE=Ø);
     RNAME=' ';
     IF RSHIP='1' & I1=Ø
     THEN DO;
          RNAME=' PARENT TABLE(S):';
          I1=1;
     END;
     IF RSHIP='2' & I2=Ø
     THEN DO;
          RNAME=' CHILD  TABLE(S):';
          I2=1;
     END;
     IF DELETERULE='R' THEN DNAME='RESTRICT';
     IF DELETERULE='C' THEN DNAME='CASCADE';
     IF DELETERULE='N' THEN DNAME='SET NULL';
     PUT SKIP LIST (RNAME||' '||CREATORR||' '||SUBSTR(TBNAMER,1,18)
                    ||' '||RELNAME||' '||DNAME);
     EXEC SQL FETCH C4 INTO
       :RSHIP, :CREATORR, :TBNAMER, :RELNAME, :DELETERULE;
  END;
  EXEC SQL CLOSE C4;
 END PTINFO5;
```

Bernard Zver
Database Administrator
Informatika Maribor (Slovenia)

Extracting catalog information

The following REXX program extracts catalog information from DB2 and generates DDL statements for DB2 objects. Four members are included – a REXX member, a JCL member, a panel member, and an output member.

JCL

```
//CREDDLJ    JOB (ACCT(tm)),,CLASS=P,MSGCLASS=X,MSGLEVEL=(1,1)
//*
//DBPØ      EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø,REGION=4Ø96K
//STEPLIB  DD   DSN=ISP.SISPLOAD,DISP=SHR
//         DD   DSN=PDSN.SDSNEXIT,DISP=SHR
//         DD   DSN=PDSN.SDSNLOAD,DISP=SHR
//SYSEXEC  DD   DSN=SYSPDBA.REXXLIB,DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
 PROFILE NOPREFIX
 %CREDDLR P  PAAAAAAA PZZZZZZZ BATCH
/*
```

REXX EXEC

```
CREDDLR :

/* REXX                                                   */
/**********************************************************/
/* Description and usage :                                */
/* This program creates DDL members for DB2 objects.      */
/* It creates DDL members for tablespace, tables that     */
/* are created in that tablespace, related indexes,       */
/* primary key, foreign keys, and privilege statements.   */
/*   Member name is given as tablespace name.             */
/* This program is developed on DB2 V51Ø. DSNTIAUL unload */
/* program is used for selecting rows from catalog tables.*/
/*                                                        */
/* This program can be run batch or interactive mode.     */
/*                                                        */
/* 1. batch mode :                                        */
/*              a. All members are created by this        */
/*                 program on 5th and 2Øth days of each  */
/*                 month.                                 */
/*              b. Changed objects members are created    */
/*                 by this program on other days.         */
/*               You have to give all arguments in batch  */
/*               mode like this.                          */
/*               wdbid = DB2 subsystem-id (P:production   */
/*                                         T:test         */
/*                                         D:development  */
/*               wts1  = beginning tablespace range       */
/*               wts2  = last tablespace range            */
/*               ws_all= 'BATCH' for batch mode           */
/*                                                        */
/* 2. Interactive mode: This mode creates a specific      */
/*                      tablespace's DDL member, given    */
/*                      from the panel.                   */
/*                      In this mode, program must call   */
/*                      with no arguments.                */
/* End of the program, index member is created which is   */
/* very useful in finding DDL member for tables.          */
/**********************************************************/
arg wdbid wts1 wts2 ws_all
   wuser = sysvar(sysuid)
   wdate = date('E')
   wtime = time()
   wday = substr(wdate,1,2)
   whour = Ø
   whour = substr(wtime,1,2)
   if wday = 'Ø6' then
      if whour < 2Ø then
           wday = 'Ø5'
   if wday = '21' then
      if whour < 2Ø then
           wday = '2Ø'
   if ws_all = ' ' then partial = 'XXXX'
   if ws_all = 'BATCH' then
   if wdbid  = 'D' | wdbid = 'P' then
   if wday = 'Ø5' | wday = '2Ø' then do
      delete syspdba.pdØ.ddl.old
      rename 'syspdba.pdØ.ddl','syspdba.pdØ.ddl.old'
      alloc da(syspdba.pdØ.ddl) new cylinders space(3Ø,1Ø) dir(2ØØ),
         lrecl(8Ø) blksize(132ØØ) dsorg(po) recfm(f,b)
   end
   if ws_all = 'BATCH' then
   if wday = 'Ø5' | wday = '2Ø' then do
      partial = 'FULL'
      ws_all  = 'ALL'
   end
   if wdbid = ' ' then do
      ispexec libdef ispplib dataset id('syspdba.panellib')
      ispexec display panel(creddlp)
      if rc ¬= Ø then return
      if d = 'D' | d = 'T' | d = 'P' then
                    nop
```

```
            else
            return
    wdbid = d
    wts1 = tsname1
    wts2 = tsname1
    end
  dbid = wdbid
  i = Ø
  s = Ø
  ts = Ø
  tb = Ø
  rr = Ø
  ts_num = Ø
  tb_num = Ø
  tsp_num = Ø
  cl_num = Ø
  ix_num = Ø
  ixp_num = Ø
  ixk_num = Ø
  hsts = Ø
  wflag = 'Ø'
  ws_input.1 = ' '
  wtsname1 = wts1
  wtsname2 = wts2
  call select_tablespace
  if ws_all = 'ALL' then
           call cre_index
  return
/*****************************************************************/
/*   select_tablespace                                           */
/*****************************************************************/
select_tablespace:
  if partial = 'PARTIAL' then do
     s= 29
     sel.1 = ' select a.name,a.creator,a.dbname,'
     sel.2 = ' a.bpool,digits(a.partitions),'
     sel.3 = ' a.lockrule,a.eraserule,digits(a.ntables),a.closerule,'
     sel.4 = ' digits(a.segsize),digits(a.lockmax),a.type,a.lockpart '
     sel.5 = ' from sysibm.systablespace a,sysibm.systables b ,'
     sel.6 = ' sysibm.sysindexes c'
     sel.7 = 'where  a.name = b.tsname  and b.name = c.tbname and'
     sel.8 = ' b.creator = c.tbcreator     '
     sel.9 = 'and (    (date(a.createdts) > current date - 2 days)'
     sel.1Ø= '     or  (date(a.alteredts) > current date - 2 days)'
     sel.11= '     or  (date(b.createdts) > current date - 2 days)'
     sel.12= '     or  (date(b.alteredts) > current date - 2 days)'
     sel.13= '     or  (date(c.createdts) > current date - 2 days)'
     sel.14= '     or  (date(c.alteredts) > current date - 2 days)'
     sel.15= '    )'
     sel.16= '  union '
     sel.17= ' select a.name,a.creator,a.dbname,'
     sel.18= ' a.bpool,digits(a.partitions),'
     sel.19= ' a.lockrule,a.eraserule,digits(a.ntables),a.closerule,'
     sel.2Ø= ' digits(a.segsize),digits(a.lockmax),a.type,a.lockpart '
     sel.21= ' from sysibm.systablespace a,sysibm.systables b ,'
     sel.22= ' sysibm.sysindexes c'
     sel.23= 'where  a.name = b.tsname  and b.name = c.tbname    '
     sel.24= 'and (    (date(a.createdts) > current date - 2 days)'
     sel.25= '     or  (date(a.alteredts) > current date - 2 days)'
     sel.26= '     or  (date(b.createdts) > current date - 2 days)'
     sel.27= '     or  (date(b.alteredts) > current date - 2 days)'
     sel.28= '    )'
     sel.29= ' order by 1  with ur       ; '
 end
 else  do
      s= 9
      sel.1 = ' select name,creator,dbname,'
      sel.2 = ' bpool,digits(partitions),'
      sel.3 = ' lockrule,eraserule,digits(ntables),closerule,'
      sel.4 = ' digits(segsize),digits(lockmax),type,lockpart'
      sel.5 = ' from sysibm.systablespace '
      sel.6 = 'where  name      between '
      sel.7 = '  ''' || wtsname1  || ''''  || ' and '
      sel.8 = '  ''' || wtsname2  || ''''
      sel.9 = ' order by name with ur     ; '
 end
 wflag = 'Ø'
 call db2_call
 ts_num = hsts
 do i = 1 to ts_num
         ts_name.i        =  substr(sel_out.i,1,8)
         ts_creator.i     =  substr(sel_out.i,9,8)
         ts_dbname.i      =  substr(sel_out.i,17,8)
         ts_bpool.i       =  substr(sel_out.i,25,8)
         ts_part.i        =  substr(sel_out.i,33,5)
         ts_lock.i        =  substr(sel_out.i,38,1)
         ts_erase.i       =  substr(sel_out.i,39,1)
         ts_ntables.i     =  substr(sel_out.i,4Ø,5)
         ts_close.i       =  substr(sel_out.i,45,1)
         ts_segsize.i     =  substr(sel_out.i,46,5)
         ts_lockmax.i     =  substr(sel_out.i,51,1Ø)
         ts_type.i        =  substr(sel_out.i,61,1)
         ts_lockpart.i    =  substr(sel_out.i,62,1)
 end
 do ts = 1 to ts_num
      wi = Ø
      wtsname = ts_name.ts
      ws_ddl_type = 'ts1'
      call ddl_member_input
      call select_tablespacepart
```

```
       ws_ddl_type = 'ts2'
       call ddl_member_input
       call select_tables
       call cre_ddl_member
  end
  return
/***************************************************************/
/*   select_tablespacepart                                     */
/***************************************************************/
select_tablespacepart:
  s= 9
  sel.1 = ' select '
  sel.2 = ' digits(partition),digits(pqty),digits(sqty),'
  sel.3 = ' stortype,storname,vcatname,'
  sel.4 = ' substr(limitkey,1,3Ø),digits(freepage),'
  sel.5 = ' digits(pctfree),compress,gbpcache'
  sel.6 = ' from sysibm.systablepart  '
  sel.7 = ' where tsname   = '
  sel.8 = '  ''' || wtsname   || ''''
  sel.9 = ' order by 1  with ur    ; '
  wflag = 'Ø'
  call db2_call
  tsp_num = hsts
  do i = 1 to tsp_num
      tsp_partition.i = substr(sel_out.i,1,5)
      tsp_pqty.i      = substr(sel_out.i,6,1Ø)
      tsp_sqty.i      = substr(sel_out.i,16,5)
      tsp_sttype.i    = substr(sel_out.i,21,1)
      tsp_storname.i  = substr(sel_out.i,22,8)
      tsp_vcatname.i  = substr(sel_out.i,3Ø,8)
      tsp_limitkey.i  = substr(sel_out.i,38,3Ø)
      tsp_freepage.i  = substr(sel_out.i,68,5)
      tsp_pctfree.i   = substr(sel_out.i,73,5)
      tsp_compress.i  = substr(sel_out.i,78,1)
      tsp_gbpcache.i  = substr(sel_out.i,79,1)
  end
  tsp = tsp_num
  ws_ddl_type = 'tsp'
  call ddl_member_input
  return
/***************************************************************/
/*   select_tables                                             */
/***************************************************************/
select_tables:
  ws_type = 'T'
  s= 9
  sel.1 = ' select '
  sel.2 = ' name,creator,datacapture,edproc,valproc,clustertype,'
  sel.3 = ' auditing'
  sel.4 = ' from sysibm.systables  '
  sel.5 = ' where tsname   =  '
  sel.6 = '  ''' || wtsname   || ''''
  sel.7 = '  and type =  '
  sel.8 = '  ''' || ws_type   || ''''
  sel.9 = ' order by 1  with ur    ; '
  wflag = 'Ø'
  call db2_call
  tb_num = hsts
  do i = 1 to tb_num
      tb_name.i       = substr(sel_out.i,3,18)
      tb_creator.i    = substr(sel_out.i,21,8)
      tb_datacap.i    = substr(sel_out.i,29,1)
      tb_edproc.i     = substr(sel_out.i,3Ø,8)
      tb_valproc.i    = substr(sel_out.i,38,8)
      tb_clstype.i    = substr(sel_out.i,46,1)
      tb_auditing.i   = substr(sel_out.i,47,1)
  end
  do tb = 1 to tb_num
      wtbname    = translate(tb_name.tb,'4Ø'x,'ØØ'x)
      wtbcreator = translate(tb_creator.tb,'4Ø'x,'ØØ'x)
      rr = rr + 1
      ws_index.rr = ts_dbname.ts || '  ' || ts_name.ts || '     '
      ws_index.rr = ws_index.rr || wtbcreator || '.' || wtbname
      ws_ddl_type = 'tb'
      call ddl_member_input
      call select_columns
      call select_indexes
      call select_tabauth
  end
  return
/***************************************************************/
/*   select_columns                                            */
/***************************************************************/
select_columns:
  s= 9
  sel.1 = ' select '
  sel.2 = ' name,digits(colno),coltype,digits(length),'
  sel.3 = ' digits(scale),nulls,default,digits(keyseq)'
  sel.4 = ' from sysibm.syscolumns '
  sel.5 = ' where tbname   =  '
  sel.6 = '  ''' || wtbname   || ''''
  sel.7 = ' and  tbcreator =  '
  sel.8 = '  ''' || wtbcreator || ''''
  sel.9 = ' order by 2  with ur    ; '
  wflag = 'Ø'
  call db2_call
  cl_num = hsts
  do i = 1 to cl_num
      cl_name.i       = substr(sel_out.i,3,18)
      cl_coltype.i    = substr(sel_out.i,26,8)
```

```
        cl_length.i     = substr(sel_out.i,34,5)
        cl_scale.i      = substr(sel_out.i,39,5)
        cl_nulls.i      = substr(sel_out.i,44,1)
        cl_default.i    = substr(sel_out.i,45,1)
        cl_keyseq.i     = substr(sel_out.i,46,5)
    end
    s=  9
    sel.1 = ' select '
    sel.2 = ' relname,reftbcreator,reftbname,'
    sel.3 = ' deleterule,digits(colcount)'
    sel.4 = ' from sysibm.sysrels  '
    sel.5 = ' where tbname   =  '
    sel.6 = '  ''' || wtbname   || ''''
    sel.7 = ' and  creator =  '
    sel.8 = '  ''' || wtbcreator || ''''
    sel.9 = ' order by 1   with ur    ; '
    wflag = 'Ø'
    call db2_call
    re_num = hsts
    do i = 1 to re_num
        re_relname.i    = substr(sel_out.i,1,8)
        re_refcre.i     = substr(sel_out.i,9,8)
        re_reftab.i     = substr(sel_out.i,19,18)
        re_delrule.i    = substr(sel_out.i,37,1)
        re_colcount.i   = substr(sel_out.i,38,5)
    end
        ws_ks = Ø
        ws_pri_flag = 'Ø'
        ws_ddl_type = 'cl'
        call ddl_member_input
    return
/*****************************************************************/
/*   select_indexes                                              */
/*****************************************************************/
select_indexes:
    s= 9
    sel.1 = ' select '
    sel.2 = ' name,creator,uniquerule,clustering,indexspace,'
    sel.3 = ' bpool,eraserule,closerule,indextype'
    sel.4 = ' from sysibm.sysindexes '
    sel.5 = ' where tbname   =  '
    sel.6 = '  ''' || wtbname   || ''''
    sel.7 = ' and  tbcreator =  '
    sel.8 = '  ''' || wtbcreator || ''''
    sel.9 = ' order by 2,1 with ur   ; '
    wflag = 'Ø'
    call db2_call
    ix_num = hsts
    do i = 1 to ix_num
        ix_name.i       = substr(sel_out.i,3,18)
```

```
      ix_creator.i    = substr(sel_out.i,21,8)
      ix_uniquerule.i = substr(sel_out.i,29,1)
      ix_clustering.i = substr(sel_out.i,3Ø,1)
      ix_ixspace.i    = substr(sel_out.i,31,8)
      ix_bpool.i      = substr(sel_out.i,39,8)
      ix_eraserule.i  = substr(sel_out.i,47,1)
      ix_closerule.i  = substr(sel_out.i,48,1)
      ix_indextype.i  = substr(sel_out.i,49,1)
  end
  do ix = 1 to ix_num
      ws_ddl_type = 'ix1'
      call ddl_member_input
      wixname  = translate(ix_name.ix,'4Ø'x,'ØØ'x)
      wixcreator = translate(ix_creator.ix,'4Ø'x,'ØØ'x)
      call select_keys
      call select_indexpart
      ws_ddl_type = 'ix2'
      call ddl_member_input
      ws_ddl_type = 'ix3'
      call ddl_member_input
  end
  return
/***************************************************************/
/*   select_keys                                               */
/***************************************************************/
select_keys:
  s= 8
  sel.1 = ' select '
  sel.2 = ' colseq,colname,ordering'
  sel.3 = ' from sysibm.syskeys      '
  sel.4 = ' where ixname   =  '
  sel.5 = '  ''' || wixname   || ''''
  sel.6 = ' and  ixcreator =  '
  sel.7 = '  ''' || wixcreator || ''''
  sel.8 = ' order by 1  with ur    ; '
  wflag = 'Ø'
  call db2_call
  ixk_num = hsts
  do i = 1 to ixk_num
      ixk_colname.i    = substr(sel_out.i,5,18)
      ixk_ordering.i   = substr(sel_out.i,23,1)
  end
  ws_ddl_type = 'ixk'
  call ddl_member_input
  return
/***************************************************************/
/*   select_indexpart                                          */
/***************************************************************/
select_indexpart:
  s= 1Ø
  sel.1 = ' select '
  sel.2 = ' digits(partition),digits(pqty),digits(sqty),storname,'
  sel.3 = ' vcatname,digits(freepage),digits(pctfree),gbpcache,'
  sel.4 = ' substr(limitkey,1,40)'
  sel.5 = ' from sysibm.sysindexpart'
  sel.6 = ' where ixname   =  '
  sel.7 = '  ''' || wixname   || ''''
  sel.8 = ' and  ixcreator =  '
  sel.9 = '  ''' || wixcreator || ''''
  sel.10= ' order by 1  with ur    ; '
  wflag = '0'
  call db2_call
  ixp_num = hsts
  do i = 1 to ixp_num
      ixp_partition.i  = substr(sel_out.i,1,5)
      ixp_pqty.i       = substr(sel_out.i,6,10)
      ixp_sqty.i       = substr(sel_out.i,16,5)
      ixp_storname.i   = substr(sel_out.i,21,8)
      ixp_vcatname.i   = substr(sel_out.i,29,8)
      ixp_freepage.i   = substr(sel_out.i,37,5)
      ixp_pctfree.i    = substr(sel_out.i,42,5)
      ixp_gbpcache.i   = substr(sel_out.i,47,1)
      ixp_limitkey.i   = substr(sel_out.i,48,40)
  end
  return
/*****************************************************************/
/*   select_tableauth                                            */
/*****************************************************************/
select_tabauth:
  s= 12
  sel.1 = ' select '
  sel.2 = ' grantee,alterauth,deleteauth,indexauth,'
  sel.3 = ' insertauth,selectauth,updateauth'
  sel.4 = ' from sysibm.systabauth  '
  sel.5 = ' where tcreator =  '
  sel.6 = '  ''' || wtbcreator || ''''
  sel.7 = ' and  ttname    =  '
  sel.8 = '  ''' || wtbname || ''''
  sel.9 = ' and  grantee  ¬=  '
  sel.10= '  ''' || wtbcreator || ''''
  sel.11= ' and  granteetype ¬= ' || '''' || 'P' || ''''
  sel.12= ' order by 1  with ur    ; '
  wflag = '0'
  call db2_call
  tba_num = hsts
  do i = 1 to tba_num
      tba_grantee.i    = substr(sel_out.i,1,8)
      tba_alter.i      = substr(sel_out.i,9,1)
      tba_delete.i     = substr(sel_out.i,10,1)
      tba_index.i      = substr(sel_out.i,11,1)
      tba_insert.i     = substr(sel_out.i,12,1)
      tba_select.i     = substr(sel_out.i,13,1)
```

```
        tba_update.i      = substr(sel_out.i,14,1)
    end
        ws_ddl_type = 'gr'
        call ddl_member_input
    return
/*************************************************************/
/*   select_foreign_key columns for dependent table          */
/*************************************************************/
find_foreign_key:
    s= 1Ø
    sel.1 = ' select '
    sel.2 = ' colname,digits(colseq)'
    sel.3 = ' from sysibm.sysforeignkeys'
    sel.4 = ' where tbname   =  '
    sel.5 = '  ''' || wtbname  || ''''
    sel.6 = ' and  creator =  '
    sel.7 = '  ''' || wtbcreator || ''''
    sel.8 = ' and  relname =  '
    sel.9 = '  ''' || wfk_relname || ''''
    sel.1Ø= ' order by 2  with ur    ; '
    wflag = 'Ø'
    call db2_call
    fk_num = hsts
    do i = 1 to fk_num
        fk_colname.i    = substr(sel_out.i,3,18)
    end
    s=  11
    sel.1 = ' select '
    sel.2 = ' b.colname,b.colseq    '
    sel.3 = ' from sysibm.sysindexes a,sysibm.syskeys b'
    sel.4 = ' where a.tbcreator = '
    sel.5 = '  ''' || wpk_creator || ''''
    sel.6 = ' and  a.tbname =  '
    sel.7 = '  ''' || wpk_tbname || ''''
    sel.8 = ' and  a.uniquerule = '  || '''' || 'P' || ''''
    sel.9 = ' and  a.creator =  b.ixcreator '
    sel.1Ø= ' and  a.name = b.ixname '
    sel.11= ' order by 2   with ur    ; '
    wflag = 'Ø'
    call db2_call
    pk_num = hsts
    do i = 1 to pk_num
        pk_colname.i    = substr(sel_out.i,3,18)
    end
    return
```

Editor's note: this article will be concluded next month.

Ali Ozturk
DBA
Pamukbank (Turkey)

DB2 news

IBM has announced Version 6.1 of DB2 Satellite Edition for Windows and DB2 Query Patroller for NT.

The DB2 Satellite product provides a satellite database system, which is transparent to the end user, bi-directional update-anywhere replication between satellite systems and any DB2 server, and the facility for mass roll-outs of satellite systems through DB2 install enhancements.

DB2 Intelligent Miner 6.1 has new analysis, data exploration, and scalability options while making critical data more accessible. Enhancements include integration with warehouse and business intelligence tools.

IBM has also announced Version 2.1.1 of its DB2 DataJoiner, now with support for Sun Solaris – adding to AIX and NT – and access to NCR's Teradata via DataJoiner on AIX.

The new version provides integrated data replication. DataPropagator is integrated in DataJoiner providing both synchronous and asynchronous data access. DataJoiner provides continuous updates as well as point-in-time replication along with integrated replication administration.

For further information contact your local IBM representative.

* * *

Novera Software has announced Release 4.6 of its Novera e-business integration and management pack, with extended integration capabilities for legacy systems and Web applications through support for DB2, MQSeries, Oracle8*i*, and Allaire ColdFusion.

Users can view, create, delete, configure, manage, and monitor message queues. Developers can access DB2 and Oracle8i data.

For further information contact:
Novera Software, 3 Burlington Woods Drive, Burlington, MA 01803, USA.
Tel: (781) 270 4422.
URL: http://www.novera.com.

* * *

Princeton Softech has announced that its SyncPoint software now enables Windows CE users to synchronize an enterprise database with a local database on handheld or palm-sized PCs.

SyncPoint isolates slices of the corporate database and distributes the data to remote users, with an administrative console that provides real-time feedback on the synchronization process.

It's said to maintain 100% data accuracy while segmenting, distributing, securing, and synchronizing data in a multi-vendor database and application environment.

It accommodates large numbers of users with scalable parallel processing and dynamic load balancing. It directly supports DB2 UDB, DB2 for OS/390, Oracle, Sybase, and SQL Server and has ODBC support for other client databases.

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