76

DB2

update

February 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

Disclaimer

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

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.

DB2 Version 5 SQL enhancements

Although DB2 for OS/390 Version 5 contains numerous usability enhancements to warm the hearts of database administrators around the world, there are a handful of significant new features that programmers should know about, too. This article will cover the four SQL enhancements plus an important Optimizer improvement that DB2 programmers can put to use immediately.

CASE EXPRESSIONS

Added in support of the SQL-92 standard, CASE expressions allow for the embedding of conditional logic directly into a SELECT list or WHERE clause. Put simply, you can use CASE expressions to create conditional logic wherever an expression is allowed.

Let's assume that you've been asked to produce a list of all employees together with a description of their current status. Prior to Version 5, you might code something that requires multiple passes through the data, as follows:

```
SELECT LASTNAME, 'ACTIVE'
  FROM EMP_TAB
 WHERE STATUS = 'A'
UNION ALL
SELECT LASTNAME, 'RETIRED'
  FROM EMP_TAB
 WHERE STATUS = 'R'
UNION ALL
SELECT LASTNAME, 'TERMINATED'
  FROM EMP_TAB
 WHERE STATUS = 'T'
UNION ALL
SELECT LASTNAME, 'UNKNOWN'
  FROM EMP_TAB
 WHERE STATUS NOT IN ('A','R','T')
```

With the CASE expression at your disposal, you'll do better to satisfy those same user requirements with a single pass through the data:

```
SELECT LASTNAME,
       CASE STATUS
            WHEN 'A' THEN 'ACTIVE'
            WHEN 'R' THEN 'RETIRED'
            WHEN 'T' THEN 'TERMINATED'
```

```
            ELSE 'UNKNOWN'
       END
FROM EMP_TAB
```

Now let's assume you've been asked to show a list of every employee whose commission is at least double his or her salary. The most intuitive solution happens to be fraught with a potential division by zero error:

```
SELECT LASTNAME, DEPTNAME
FROM EMP_TAB
WHERE (COMMISSION/SALARY) > 2
```

Using a CASE expression, the potential for division by zero is eliminated:

```
SELECT LASTNAME, DEPTNAME
FROM EMP_TAB
WHERE (CASE WHEN SALARY = 0 THEN 0
            ELSE COMMISSION/SALARY END) > 2
```

TWO NEW SCALAR FUNCTIONS

IBM has added two new scalar functions to bring the total to 24. The first new scalar is STRIP. It is used to remove leading, trailing, or both leading *and* trailing characters from a larger character string.

This somewhat contrived code example removes leading blanks from last name, trailing zeros from employee number, and both leading and trailing zeros from the character representation of the salary:

```
SELECT STRIP(LASTNAME,L,' '),
       STRIP(EMPNO,T,'0'),
       STRIP(DIGITS(SALARY),B,'0')
FROM EMP_TAB
```

Note that the first argument of the STRIP function must be a character string, hence the need for the DIGITS function. If you specify a second argument, it indicates whether characters are removed from the end or beginning of the string. DB2 gives us the flexibility to code the second argument spelled out (LEADING, TRAILING, BOTH) or abbreviated by its first letter as shown in the previous code fragment. If you do not specify a second argument, blanks are removed from both the end and the beginning of the string. The third value is optional; it defaults to a blank. Also, if the third value contains more than one character, an error is returned.

The second new scalar is NULLIF. This function returns null if the two arguments are equal; otherwise, it returns the value of the first argument. NULLIF is the inverse of the VALUE and COALESCE functions. Although it might not be as useful as some of the other scalars, there are certain problems that can be solved more easily through the use of NULLIF. For example, let's assume you've been asked to find the company-wide employee salary, but the salary column doesn't allow NULLs, and zeros have been used to represent missing data. The intuitive solution uses the well-known AVG function:

```
SELECT AVG(SALARY)
  FROM EMP_TAB
```

Using this technique, the presence of any artificial zeros will skew the average. Perhaps a better solution is to nest the NULLIF function within the AVG function:

```
SELECT AVG(NULLIF(SALARY,0))
  FROM EMP_TAB
```

LOCK TABLE ENHANCEMENT

In keeping with IBM's long term emphasis on managing partitions independently, the LOCK TABLE statement has been extended to support the PART option. For example, suppose the employee table is partitioned by department number, and all employees in department 002 are being given a 10% pay rise for recent outstanding work. You write a one-time batch update program to implement the rise that starts with this statement:

```
EXEC SQL
  LOCK TABLE EMP_TAB PART 2 IN EXCLUSIVE MODE
END-EXEC.
```

For this to work, EMP_TAB must belong to a partitioned tablespace that is defined using the LOCKPART YES option. Also added for Version 5, the LOCKPART option on the CREATE TABLESPACE statement enables selective partition locking.

NOTEWORTHY OPTIMIZER ENHANCEMENTS

Suppose you've been asked to list all employees hired within the last month. Knowing the strong date processing capabilities of DB2, you

decide to code something very intuitive:

```
SELECT LASTNAME, FIRSTNAME, HIREDATE
FROM EMP_TAB
WHERE HIREDATE > CURRENT DATE - 1 MONTH
```

You know there's an index on HIREDATE, so you are very surprised when the statement runs for nearly one minute elapsed time. A quick look at the EXPLAIN output indicates that the index on HIREDATE is not being used. The problem: prior to Version 5, predicates that contain non-column expressions were not evaluated until stage 2. The common solution: programmers needed a separate step to obtain the month-old date, typically by selecting from a one-row table set up by a DBA expressly for date manipulation.

```
SELECT CURRENT DATE - 1 MONTH
  INTO :TEMP-DATE
FROM ONE_ROW_TABLE

SELECT LASTNAME, FIRSTNAME, HIREDATE
FROM EMP_TAB
WHERE HIREDATE > :TEMP-DATE
```

New for Version 5, predicates of the form COL op non-col_expr can be evaluated at stage 1, and may also be indexable, which improves the performance of any queries that use those predicates, including the commonly used date arithmetic shown in the former example.

Coincidentally, new for Version 5, IBM has added a one-row table called SYSIBM.SYSDUMMY1 to the catalog for the same reason that many DBAs have set up their own one-row tables. Ironically, the Optimizer improvement discussed above largely, but not completely, eliminates the need for such a table!

LOOKING AHEAD

With Version 5 now over a year old, it's time to begin looking to the exciting new features that await us in Version 6, including user-defined functions, triggers, and many more new scalar functions. But that, as they say, is another story.

John T Bradford
Greenbrier and Russel (USA)

A flexible data copy facility

INTRODUCTION

Requests for moving data from tables in one database to tables in another seem more common now than ever. I have developed functions (a couple of them previously published in *DB2 Update*) to copy a single table from a production subsystem to a test subsystem, and to copy an entire database. New uses for DB2 as a data warehousing platform and as an integral part of multi-platform distributed applications have resulted in more frequent requests to copy between different databases and different subsystems. To satisfy these frequent requests, I needed a fully flexible function that would allow me to choose one or more tables from a list and automatically generate the JCL to copy the data in them to another database, possibly on another subsystem. Described here is the DB2COPY function I have developed to satisfy this need.

DB2COPY FUNCTION

DB2COPY is invoked through an ISPF panel. The user specifies the source and target subsystems, databases, table creator names, and a table name. All except the subsystem values may be wild-carded (with the valid SQL character '%') or simply left blank (to indicate all).

After the panel fields have been entered and the user presses ENTER, a selection list on the bottom half of the panel will be displayed listing the qualifying tables for copying. The only qualification is that, given the source and target subsystems, databases, and creators, the table names must match between the two.

At this point, the user types an 'S' on the line next to the tables that need to be copied. Alternatively, the ALL command will automatically select all tables in the list. The RES command will deselect all the tables.

Once the correct tables have been selected, either the GEN or GENS command will generate JCL to perform the copy. The difference between the two is that GENS unloads all the tables in a single step and

loads them into the target database in a single step. Because of the limitations of the DSNTIAUL unload program, a maximum of 100 tables can be unloaded at a time. In the case of copying more than 100 tables, the GEN command will generate JCL that unloads and loads each table in a single step. When the JCL exceeds 255 steps, it is broken into multiple jobs.

DB2COPY REXX EXEC

The REXX EXEC DB2COPY displays the panel and generates the JCL that performs the data copy. Open Software Technologies' REXXTOOLS/MVS is used to connect to DB2 and access the system catalogs.

DB2COPYX REXX EXEC

The DB2COPYX REXX EXEC is a simple function that replaces some of the load control keywords in the SYSPUNCH generated by the DSNTIAUL unload program. This implementation uses the REPLACE and ENFORCE NO keywords.

THE GENERATED JCL

The IRXJCL program is used to execute the DB2COPYX REXX EXEC in the JCL. It is described in the *TSO/E REXX/MVS User's Guide*. IKJEFT01 could have been used, but since the REXX EXECs do not require TSO/E services, the IRXJCL program is a suitable choice. The IKJEFT01 program creates a TSO address space, IRXJCL does not.

CONCLUSION

This function has been a big time-saver. Its flexibility has made it the most commonly used method of copying DB2 data at my shop.

PDB2COPY ISPF PANEL

```
)ATTR
/*********************************************************************/
/*                                                                   */
```

```
/* PDB2COPY - COPY DB2 TABLE DATA FROM ANY SUBSYSTEM TO ANY OTHER     */
/*                                                                    */
/**********************************************************************/
   + TYPE(TEXT)   INTENS(LOW)   COLOR(BLUE)         SKIP(ON)
   % TYPE(TEXT)   INTENS(HIGH)  COLOR(WHITE)        SKIP(ON)
   $ TYPE(TEXT)   INTENS(HIGH)  COLOR(RED)          SKIP(ON)
   ! TYPE(TEXT)   INTENS(HIGH)  COLOR(YELLOW)       SKIP(ON)
   # TYPE(OUTPUT) INTENS(HIGH)  COLOR(TURQUOISE)    CAPS(ON)
   @ TYPE(OUTPUT) INTENS(HIGH)  COLOR(YELLOW)       CAPS(ON)
)BODY CMD(C)
%--------------- DB2 DATA COPY FUNCTION ----------------
%COMMAND ===>_C
%SCR->_AMT +
+
%Source tables+           %Target tables+        %Commands+
+Subsystem ===>_SDSS+     +Subsystem ===>_TDSS+  !GEN+ Generate JCL+
+Database ====>_SRCDB  + +Database ====>_TGTDB + !ALL+ Select all rows+
+Table Owner =>_SRCOWN + +Table Owner =>_TGTOWN+ !RES+ De-Sel all rows+
+Table Name ==>_TABNAM   +                      !GENS+1-step unld/load+
+
+Output library%======>_JDSN                       + Member%===>_JMEM
+
+
%        S O U R C E           T A R G E T         TABLE
% CMD    DATABASE  OWNER       DATABASE  OWNER     NAME
%-------------------------------------+
)MODEL
_S+@T+  #SDB     +#SRCO    +  #TDB     +#TGTO    +#TABNM            +
)INIT
  .CURSOR = C
  &C = ' '
)PROC
  VER (&C LIST,GEN,GENS,END,ALL,RES,' ')
  VER (&SDSS NONBLANK)
  VER (&TDSS NONBLANK)
  VER (&JDSN NONBLANK)
  VER (&JMEM NONBLANK NAME)
  VER (&SDSS LIST,DSN1,DSNT,DSNP)
  VER (&TDSS LIST,DSN1,DSNT,DSNP)
)END
```

DB2COPY REXX EXEC

```
/* REXX ***************************************************************/
/*                                                                    */
/* DB2COPY - COPY DB2 DATA BETWEEN LIKE TABLES.                       */
/*           SOURCE AND TARGET TABLES MAY BE IN DIFFERENT SUBSYSTEMS  */
/*           OR DATABASES AND MAY HAVE DIFFERENT TABLE NAMES.         */
/*                                                                    */
/********************************************************************* REXX */
  address "ISPEXEC" "TBCREATE TBLIST"||,
```

```
    " NAMES(SDB SRCO TDB TGTO TABNM CYLP CYLX TTS T)"||,
    " NOWRITE REPLACE"
address "ISPEXEC" "VGET (ZCSDSS ZCSRCDB ZCSRCOWN) PROFILE"
address "ISPEXEC" "VGET (ZCTDSS ZCTGTDB ZCTGTOWN) PROFILE"
address "ISPEXEC" "VGET (ZCTABNAM) PROFILE"
address "ISPEXEC" "VGET (ZCJDSN ZCJMEM) PROFILE"
SDSS   = ZCSDSS
SRCDB  = ZCSRCDB
SRCOWN = ZCSRCOWN
TDSS   = ZCTDSS
TGTDB  = ZCTGTDB
TGTOWN = ZCTGTOWN
TABNAM = ZCTABNAM
JDSN   = ZCJDSN
JMEM   = ZCJMEM
redispl = 'y'
savetop = Ø
do forever
  S = ''
  address "ISPEXEC" "TBTOP TBLIST"
  address "ISPEXEC" "TBSKIP TBLIST NUMBER("savetop")"
  address "ISPEXEC" "TBDISPL TBLIST PANEL(PDB2COPY)"
  if rc > 4 then
    do
      address "ISPEXEC" "TBEND TBLIST"
      exit
    end
  savetop = ZTDTOP
  if (ZTDSELS > Ø) then
    do
      SELCOUNT = ZTDSELS
      do J = 1 TO SELCOUNT
        address "ISPEXEC" "TBPUT TBLIST"
        if T = ' ' then
          T = 'S'
        else
          T = ' '
        address "ISPEXEC" "TBPUT TBLIST"
        if J < SELCOUNT then
          address "ISPEXEC" "TBDISPL TBLIST"
      end
      ITERATE
    end
  if C = 'ALL' then
    do
      address "ISPEXEC" "TBTOP TBLIST"
      do while (rc = Ø)
        T = 'S'
        address "ISPEXEC" "TBPUT TBLIST"
        address "ISPEXEC" "TBSKIP TBLIST"
      end
```

```
        ITERATE
      end
    if C = 'RES' then
      do
        address "ISPEXEC" "TBTOP TBLIST"
        do while (rc = Ø)
          T = ' '
          address "ISPEXEC" "TBPUT TBLIST"
          address "ISPEXEC" "TBSKIP TBLIST"
        end
        ITERATE
      end
    if C = 'GEN' then
      do
        call PØØØ4_BUILD_JOB
        ITERATE
      end
    if C = 'GENS' then
      do
        call PØØØ5_BUILD_JOB
        ITERATE
      end
    call PØØØØ_BUILD_DISPLAY
  end
  exit

/*****************************************************************/
/* PØØØØ_BUILD_DISPLAY                                           */
/*****************************************************************/
PØØØØ_BUILD_DISPLAY:
     editok = 'n'
     tgtcnt = Ø
     srccnt = Ø
     if (TABNAM > '') & (TABNAM <> '%') then
       editok = 'y'
     if (TGTOWN > '') & (TGTOWN <> '%') then
       tgtcnt = tgtcnt + 1
     if (TGTDB > '') & (TGTDB <> '%') then
       tgtcnt = tgtcnt + 1
     if (SRCOWN > '') & (SRCOWN <> '%') then
       srccnt = srccnt + 1
     if (SRCDB > '') & (SRCDB <> '%') then
       srccnt = srccnt + 1
     if (tgtcnt > Ø) & (srccnt > Ø) then
       editok = 'y'
     if editok = 'n' then
       do
         ZEDSMSG = "No criteria specified"
         address "ISPEXEC" "SETMSG MSG(ISRZØØ1)"
         return
       end
```

```
      if ZCSDSS <> SDSS then
        redispl = 'y'
      if ZCSRCDB <> SRCDB then
        redispl = 'y'
      if ZCSRCOWN <> SRCOWN then
        redispl = 'y'
      if ZCTDSS <> TDSS then
        redispl = 'y'
      if ZCTGTDB <> TGTDB then
        redispl = 'y'
      if ZCTGTOWN <> TGTOWN then
        redispl = 'y'
      if ZCTABNAM <> TABNAM then
        redispl = 'y'
      if redispl = 'y' then
        do
          call P0001_BLD_SRCLIST;
          call P0002_BLD_TGTLIST;
          call P0003_MERGE_LISTS;
          ZCSDSS   = SDSS
          ZCSRCDB  = SRCDB
          ZCSRCOWN = SRCOWN
          ZCTDSS   = TDSS
          ZCTGTDB  = TGTDB
          ZCTGTOWN = TGTOWN
          ZCTABNAM = TABNAM
          ZCJDSN   = JDSN
          ZCJMEM   = JMEM
          address "ISPEXEC" "VPUT (ZCSDSS ZCSRCDB ZCSRCOWN) PROFILE"
          address "ISPEXEC" "VPUT (ZCTDSS ZCTGTDB ZCTGTOWN) PROFILE"
          address "ISPEXEC" "VPUT (ZCTABNAM) PROFILE"
          address "ISPEXEC" "VPUT (ZCJDSN ZCJMEM) PROFILE"
          redispl = 'n'
        end
return;

/*****************************************************************/
/* P0001_BLD_SRCLIST                                             */
/*****************************************************************/
P0001_BLD_SRCLIST:
    if dsnali("OPEN", SDSS, "REXXTOOL") <> 0 then do
      say "Open for plan failed.  rc="rc" REASON="reason
      exit rc
    end
    address SQL
    "EXEC SQL BEGIN DECLARE SECTION;"
    "EXEC SQL DECLARE SDBNAME  VARIABLE CHAR(8);"
    "EXEC SQL DECLARE SOWNER   VARIABLE CHAR(8);"
    "EXEC SQL DECLARE TABNAME  VARIABLE VARCHAR(18);"
    "EXEC SQL END DECLARE SECTION;"
    if SRCDB  = '' then
      SRCDB  = '%'
```

```
    if SRCOWN = '' then
      SRCOWN = '%'
    if TABNAM = '' then
      TABNAM = '%'
    TABNAME = TABNAM
    strlen = LENGTH(SRCDB)
    lastchar = substr(SRCDB,strlen,1)
    if (lastchar <> '_') & (lastchar <> '%') then
      SDBNAME = LEFT(SRCDB,8)
    else
      SDBNAME = SRCDB
    strlen = LENGTH(SRCOWN)
    lastchar = substr(SRCOWN,strlen,1)
    if (lastchar <> '_') & (lastchar <> '%') then
      SOWNER = LEFT(SRCOWN,8)
    else
      SOWNER = SRCOWN
    "EXEC SQL",
       "SELECT CREATOR, NAME, DBNAME, NPAGES",
         "FROM SYSIBM.SYSTABLES",
        "WHERE DBNAME LIKE :SDBNAME",
          "AND CREATOR LIKE :SOWNER",
          "AND NAME LIKE :TABNAME",
        "ORDER BY 2, 3, 1"
    if SQLCA.SQLCODE > 0 then
      do
        say 'DATA RETRIEVAL ERROR'
        say 'rc='||rc
        return
      end
    do i = 1 TO SQLCA.SQLROWS
      DB1.i      = DBNAME.i
      OWN1.i     = CREATOR.i
      TAB1.i     = NAME.i
      cylins     = (NPAGES.i/12+1)/15+1
      parse var cylins cyli '.' cyld
      CYLS.i     = cyli
      cylins     = cylins/10+1
      parse var cylins cyli '.' cyld
      CYLS2.i    = cyli
    end
    SRCCNT = SQLCA.SQLROWS
    if dsnali("CLOSE", "SYNC") <> 0 then do
       say "Close for plan failed.  RC="rc" REASON="reason
       exit rc
    end
return;

/*****************************************************************/
/* P0002_BLD_TGTLIST                                             */
/*****************************************************************/
```

```
P0002_BLD_TGTLIST:
    if dsnali("OPEN", TDSS, "REXXTOOL") <> 0 then do
      say "Open for plan failed.  rc="rc" REASON="reason
      exit rc
    end
    address SQL
    "EXEC SQL BEGIN DECLARE SECTION;"
    "EXEC SQL DECLARE TDBNAME  VARIABLE CHAR(8);"
    "EXEC SQL DECLARE TOWNER   VARIABLE CHAR(8);"
    "EXEC SQL DECLARE TABNAME  VARIABLE VARCHAR(18);"
    "EXEC SQL END DECLARE SECTION;"
    if TGTDB  = '' then
      TGTDB  = '%'
    if TGTOWN = '' then
      TGTOWN = '%'
    if TABNAM = '' then
      TABNAM = '%'
    TABNAME = TABNAM
    strlen = LENGTH(TGTDB)
    lastchar = substr(TGTDB,strlen,1)
    if (lastchar <> '_') & (lastchar <> '%') then
      TDBNAME = LEFT(TGTDB,8)
    else
      TDBNAME = TGTDB
    strlen = LENGTH(TGTOWN)
    lastchar = substr(TGTOWN,strlen,1)
    if (lastchar <> '_') & (lastchar <> '%') then
      TOWNER  = LEFT(TGTOWN,8)
    else
      TOWNER = TGTOWN
    TABNAME = TABNAM
    "EXEC SQL",
       "SELECT CREATOR, NAME, DBNAME, TSNAME",
         "FROM SYSIBM.SYSTABLES",
        "WHERE DBNAME LIKE :TDBNAME",
          "AND CREATOR LIKE :TOWNER",
          "AND NAME LIKE :TABNAME",
        "ORDER BY 2, 3, 1"
    if SQLCA.SQLCODE > 0 then
      do
        say 'DATA RETRIEVAL ERROR'
        say 'rc='||rc
        return
      end
    do i = 1 TO SQLCA.SQLROWS
      DB2.i      = DBNAME.i
      OWN2.i     = CREATOR.i
      TAB2.i     = NAME.i
      TS2.i      = TSNAME.i
    end
    TGTCNT = SQLCA.SQLROWS
```

```
    if dsnali("CLOSE", "SYNC") <> Ø then do
       say "Close for plan failed.  RC="rc" REASON="reason
       exit rc
    end
return;

/*****************************************************************/
/* P0003_MERGE_LISTS                                             */
/*****************************************************************/
P0003_MERGE_LISTS:
    address "ISPEXEC" "TBCREATE TBLIST"||,
        " NAMES(SDB SRCO TDB TGTO TABNM CYLP CYLX TTS T)"||,
        " NOWRITE REPLACE"
    if rc > 4 then
      do
        say 'TBCREATE ERROR '||rc
        exit
      end
    if SRCCNT = Ø then
      return
    if TGTCNT = Ø then
      return
    j = 1
    do i = 1 TO SRCCNT
      T = ' '
      SDB      = DB1.i
      SRCO     = OWN1.i
      TABNM    = TAB1.i
      CYLP     = CYLS.i
      CYLX     = CYLS2.i
      do while (TAB2.j <= TAB1.i) & (j <= TGTCNT)
        if TAB2.j = TAB1.i then
          do
            TDB      = DB2.j
            TTS      = TS2.j
            TGTO     = OWN2.j
            address "ISPEXEC" "TBADD TBLIST"
            if rc > Ø then
              do
                say 'TBADD ERROR '||rc
                address "ISPEXEC" "TBEND TBLIST"
                exit
              end
          end
        j = j + 1
      end
    end
  address "ISPEXEC" "TBTOP TBLIST"
  if rc > 4 then
    do
      say 'TBTOP ERROR '||rc
      address "ISPEXEC" "TBEND TBLIST"
```

```
      exit
    end
return;

/*****************************************************************/
/* P0004_BUILD_JOB                                               */
/*****************************************************************/
P0004_BUILD_JOB:
  tabcnt = 0
  address "ISPEXEC" "TBTOP TBLIST"
  address "ISPEXEC" "TBSKIP TBLIST"
  do while (rc = 0)
    address "ISPEXEC" "TBGET TBLIST"
    if T = 'S' then
      do
        tabcnt = tabcnt + 1
        jtabnm.tabcnt = TABNM
        jsrco.tabcnt = SRCO
        jtgto.tabcnt = TGTO
        jcylp.tabcnt = CYLP
        jcylx.tabcnt = CYLX
        jtdb.tabcnt  = TDB
        jtts.tabcnt  = TTS
      end
    T = ' '
    address "ISPEXEC" "TBPUT TBLIST"
    address "ISPEXEC" "TBSKIP TBLIST"
  end
  if tabcnt = 0 then
    return;
  stepnum = 0
  address TSO
  output_file = "'"jdsn||"("||jmem||")'"
  "ALLOC DD(OUTPUT) DSN("output_file") SHR REUSE"
  do i = 1 to tabcnt
    if stepnum = 0 then
      do
        queue "//"sysvar(sysuid)"T JOB (xxxxx,999),'DB2COPY',"
        queue "//         MSGCLASS=X,CLASS=A"
        queue "//*************************************************"
        queue "//*"
        queue "//* COPY DB2 TABLE DATA FROM ONE DATABASE TO ANOTHER."
        queue "//*"
        queue "//* GENERATED WITH THE DB2COPY FUNCTION"
        queue "//*     ON "TRANSLATE(DATE())" "TIME()" BY "USERID()
        queue "//*"
        queue "//* TABLES COPIED FROM "sdss" TO "tdss":"
        do j = 1 to tabcnt
          queue "//* "STRIP(jsrco.j)"."jtabnm.j" TO "STRIP(jtgto.j)
        end
        queue "//*"
```

```
    queue "//************************************************"
    queue "//*"
    queue "//************************************************"
    queue "//JOBLIB   DD DSN=DSNP.SDSNLOAD,DISP=SHR"
  end

stepnum = stepnum + 1
queue "//************************************************"
queue "//* UNLOAD TABLE "STRIP(jsrco.i)"."jtabnm.i
queue "//************************************************"
queue "//STEP"stepnum"  EXEC PGM=IKJEFT01,"
queue "//         DYNAMNBR=20,"
queue "//         REGION=7M,"
queue "//         TIME=1439"
queue "//SYSPRINT DD SYSOUT=*"
queue "//SYSTSPRT DD SYSOUT=*"
queue "//SYSPUNCH DD DSN=&SYSUID..DB2COPY.SYSPUNCH,"
queue "//            DISP=(NEW,CATLG,DELETE),"
queue "//            UNIT=SYSDA,"
queue "//            DCB=(LRECL=80,BLKSIZE=8000,RECFM=FB),"
queue "//            SPACE=(CYL,(1,1),RLSE)"
queue "//SYSREC00 DD DSN=&SYSUID..DB2COPY.TABLE.DATA,"
queue "//            UNIT=SYSDA,"
queue "//            DISP=(NEW,CATLG,DELETE),"
queue "//            SPACE=(CYL,("jcylp.i","jcylx.i"),RLSE)"
queue "//SYSTSIN  DD *"
queue "  DSN SYSTEM("sdss") RETRY(0) TEST(0)"
queue "  RUN PROGRAM(DSNTIAUL) PLAN(DSNTIAUL) -"
queue "  LIB('"sdss".RUNLIB.LOAD')"
queue "  END"
queue "/*"
queue "//SYSIN    DD *"
queue "  "STRIP(jsrco.i)"."jtabnm.i
queue "/*"

stepnum = stepnum + 1
queue "//************************************************"
queue "//* MODIFY LOAD CONTROL CARDS: REPLACE ENFORCE NO"
queue "//************************************************"
queue "//STEP"stepnum"  EXEC PGM=IRXJCL,"
queue "//       PARM='DB2COPYX" STRIP(jsrco.i) STRIP(jtgto.i)"'"
queue "//SYSTSPRT DD SYSOUT=*"
queue "//SYSEXEC  DD DSN=SYS1.TSO.DB2.CLIST,DISP=SHR"
queue "//INFILE   DD DSN=&SYSUID..DB2COPY.SYSPUNCH,DISP=SHR"
queue "//OUTFILE  DD DSN=&SYSUID..DB2COPY.SYSPUNCH.MODIFIED,"
queue "//            DISP=(NEW,CATLG,DELETE),"
queue "//            UNIT=SYSDA,"
queue "//            DCB=(LRECL=80,BLKSIZE=8000,RECFM=FB),"
queue "//            SPACE=(CYL,(1,1),RLSE)"

stepnum = stepnum + 1
queue "//************************************************"
```

```
queue "//* LOAD TABLE "STRIP(jtgto.i)"."jtabnm.i
queue "//*************************************************"
queue "//STEP"stepnum"  EXEC PGM=DSNUTILB,"
queue "//         COND=(4,LT),REGION=7M,"
queue "//         PARM=('"tdss"','"sysvar(sysuid)".DB2COPY','')"
queue "//STEPLIB  DD DSN=DSNP.SDSNLOAD,DISP=SHR"
queue "//DSNTRACE DD SYSOUT=*"
queue "//UTPRINT  DD SYSOUT=*"
queue "//SYSPRINT DD SYSOUT=*"
queue "//SYSUDUMP DD SYSOUT=*"
queue "//SORTOUT  DD SPACE=(CYL,(5Ø,5Ø)),UNIT=SYSDA"
queue "//SYSUT1   DD SPACE=(CYL,(5Ø,5Ø)),UNIT=SYSDA"
queue "//SYSMAP   DD SPACE=(CYL,(5Ø,5Ø)),UNIT=SYSDA"
queue "//SYSRECØØ DD DISP=SHR,DSN=&SYSUID..DB2COPY.TABLE.DATA"
queue "//SYSIN    DD DSN=&SYSUID..DB2COPY.SYSPUNCH.MODIFIED,"
queue "//            DISP=SHR"

stepnum = stepnum + 1
queue "//*************************************************"
queue "//* START FORCE TABLESPACE"
queue "//*************************************************"
queue "//STEP"stepnum"  EXEC PGM=IKJEFTØ1,DYNAMNBR=2Ø"
queue "//STEPLIB  DD DSN=DSNP.SDSNLOAD,DISP=SHR"
queue "//SYSPRINT DD SYSOUT=*"
queue "//SYSTSPRT DD SYSOUT=*"
queue "//SYSOUT   DD SYSOUT=*"
queue "//SYSUDUMP DD SYSOUT=*"
queue "//SYSTSIN  DD *"
queue "DSN SYSTEM ("tdss")"
queue "-STA DB("jtdb.i") SPACENAM("jtts.i") ACCESS(FORCE)"
queue "END"
queue "/*"

stepnum = stepnum + 1
queue "//*************************************************"
queue "//* DELETE WORK DATASETS"
queue "//*************************************************"
queue "//STEP"stepnum"  EXEC PGM=IEFBR14,COND=(4,LT)"
queue "//SYSOUT   DD SYSOUT=*"
queue "//SYSPRINT DD SYSOUT=*"
queue "//DDØ1     DD DSN=&SYSUID..DB2COPY.TABLE.DATA,"
queue "//            DISP=(OLD,DELETE,DELETE)"
queue "//DDPUNCH  DD DSN=&SYSUID..DB2COPY.SYSPUNCH,"
queue "//            DISP=(OLD,DELETE,DELETE)"
queue "//DDPUNCHM DD DSN=&SYSUID..DB2COPY.SYSPUNCH.MODIFIED,"
queue "//            DISP=(OLD,DELETE,DELETE)"

if stepnum >= 25Ø then
  do
    queue "//"
    stepnum = Ø
  end
```

```
  end
  queue "//"
  address TSO
  how_many = queued()
  "EXECIO "how_many" DISKW OUTPUT ( FINIS"
  "FREE DD(OUTPUT)"
  address "ISPEXEC"
  "LMINIT DATAID(ID1) DATASET('"jdsn"') ENQ(SHRW)"
  if rc > 0 then
    do
      ZEDSMSG = "LMINIT "||rc
      address "ISPEXEC" "SETMSG MSG(ISRZ001)"
      exit rc
    end
  "EDIT DATAID("ID1") MEMBER("jmem")"
  if rc > 4 then
    do
      ZEDSMSG = "EDIT "||rc
      address "ISPEXEC" "SETMSG MSG(ISRZ001)"
      exit rc
    end
return;

/*****************************************************************/
/* P0005_BUILD_JOB                                               */
/*****************************************************************/
P0005_BUILD_JOB:
  tabcnt = 0
  address "ISPEXEC" "TBTOP TBLIST"
  address "ISPEXEC" "TBSKIP TBLIST"
  do while (rc = 0)
    address "ISPEXEC" "TBGET TBLIST"
    if T = 'S' then
      do
        tabcnt = tabcnt + 1
        jtabnm.tabcnt = TABNM
        jsrco.tabcnt = SRCO
        jtgto.tabcnt = TGTO
        jcylp.tabcnt = CYLP
        jcylx.tabcnt = CYLX
        jtdb.tabcnt  = TDB
        jtts.tabcnt  = TTS
      end
    T = ' '
    address "ISPEXEC" "TBPUT TBLIST"
    address "ISPEXEC" "TBSKIP TBLIST"
  end
  if tabcnt = 0 then
    return;
  if tabcnt > 100 then
    do
      ZEDSMSG = "Max 100 tables for GENS"
```

```
    address "ISPEXEC" "SETMSG MSG(ISRZ001)"
    return
  end
chksrco = jsrco.1
chktgto = jtgto.1
srcosw= 'y'
tgtosw= 'y'
do i = 2 to tabcnt
  if jsrco.i <> chksrco then
    srcosw = 'n'
  if jtgto.i <> chktgto then
    tgtosw = 'n'
end
if srcosw = 'n' then
  do
    ZEDSMSG = "Src tbls have >1 owner"
    address "ISPEXEC" "SETMSG MSG(ISRZ001)"
    return
  end
if tgtosw = 'n' then
  do
    ZEDSMSG = "Tgt tbls have >1 owner"
    address "ISPEXEC" "SETMSG MSG(ISRZ001)"
    return
  end

address TSO
output_file = "'"jdsn||"("||jmem||")'"
"ALLOC DD(OUTPUT) DSN("output_file") SHR REUSE"
queue "//"sysvar(sysuid)"T JOB (xxxxx,999),'DB2COPY',"
queue "//         MSGCLASS=X,CLASS=A"
queue "//*************************************************"
queue "//*"
queue "//* COPY DB2 TABLE DATA FROM ONE DATABASE TO ANOTHER."
queue "//*"
queue "//* GENERATED WITH THE DB2COPY FUNCTION"
queue "//*     ON "TRANSLATE(DATE())" "TIME()" BY "USERID()
queue "//*"
queue "//* TABLES COPIED FROM "sdss" TO "tdss":"
do i = 1 to tabcnt
  queue "//* "STRIP(jsrco.i)"."jtabnm.i" TO "STRIP(jtgto.i)
end
queue "//*"
queue "//*************************************************"
queue "//*"
queue "//*************************************************"
queue "//JOBLIB   DD DSN=DSNP.SDSNLOAD,DISP=SHR"

queue "//*************************************************"
queue "//* UNLOAD TABLES"
queue "//*************************************************"
```

```
queue "//STEP0001 EXEC PGM=IKJEFT01,"
queue "//         DYNAMNBR=20,"
queue "//         REGION=7M,"
queue "//         TIME=1439"
queue "//SYSPRINT DD SYSOUT=*"
queue "//SYSTSPRT DD SYSOUT=*"
queue "//SYSPUNCH DD DSN=&SYSUID..DB2COPY.SYSPUNCH,"
queue "//            DISP=(NEW,CATLG,DELETE),"
queue "//            UNIT=SYSDA,"
queue "//            DCB=(LRECL=80,BLKSIZE=8000,RECFM=FB),"
queue "//            SPACE=(CYL,(1,1),RLSE)"

do i = 1 to tabcnt
  chrcnt = i - 1
  if chrcnt < 10 then chrcnt = "0"chrcnt
  queue "//SYSREC"chrcnt "DD DSN=&SYSUID..DB2COPY.TAB"chrcnt","
  queue "//            UNIT=SYSDA,"
  queue "//            DISP=(NEW,CATLG,DELETE),"
  queue "//            SPACE=(CYL,("jcylp.i","jcylx.i"),RLSE)"
end

queue "//SYSTSIN  DD *"
queue "  DSN SYSTEM("sdss") RETRY(0) TEST(0)"
queue "  RUN PROGRAM(DSNTIAUL) PLAN(DSNTIAUL) -"
queue "  LIB('"sdss".RUNLIB.LOAD')"
queue "  END"
queue "/*"
queue "//SYSIN    DD *"

do i = 1 to tabcnt
  queue "  "STRIP(jsrco.i)"."jtabnm.i
end
queue "/*"

queue "//************************************************"
queue "//* MODIFY LOAD CONTROL CARDS: REPLACE ENFORCE NO"
queue "//************************************************"
queue "//STEP0002 EXEC PGM=IRXJCL,"
queue "//         PARM='DB2COPYX" STRIP(jsrco.1) STRIP(jtgto.1)"'"
queue "//SYSTSPRT DD SYSOUT=*"
queue "//SYSEXEC  DD DSN=SYS1.TSO.DB2.CLIST,DISP=SHR"
queue "//INFILE   DD DSN=&SYSUID..DB2COPY.SYSPUNCH,DISP=SHR"
queue "//OUTFILE  DD DSN=&SYSUID..DB2COPY.SYSPUNCH.MODIFIED,"
queue "//            DISP=(NEW,CATLG,DELETE),"
queue "//            UNIT=SYSDA,"
queue "//            DCB=(LRECL=80,BLKSIZE=8000,RECFM=FB),"
queue "//            SPACE=(CYL,(1,1),RLSE)"

queue "//************************************************"
queue "//* TERMINATE UTILITY (FOR RERUNNABILITY)"
queue "//************************************************"
```

```
queue "//STEP3    EXEC PGM=IKJEFT01,DYNAMNBR=20"
queue "//STEPLIB  DD DSN=DSNP.SDSNLOAD,DISP=SHR"
queue "//SYSPRINT DD SYSOUT=*"
queue "//SYSTSPRT DD SYSOUT=*"
queue "//SYSUDUMP DD SYSOUT=*"
queue "//SYSTSIN  DD *"
queue "DSN SYSTEM ("tdss")"
queue "-TERM UTIL("sysvar(sysuid)".DB2COPY)"
queue "END"
queue "/*"

queue "//************************************************"
queue "//* LOAD TABLES"
queue "//************************************************"
queue "//STEP0003 EXEC PGM=DSNUTILB,"
queue "//         COND=(4,LT),REGION=7M,"
queue "//         PARM=('"tdss"','"sysvar(sysuid)".DB2COPY','')"
queue "//STEPLIB  DD DSN=DSNP.SDSNLOAD,DISP=SHR"
queue "//DSNTRACE DD SYSOUT=*"
queue "//UTPRINT  DD SYSOUT=*"
queue "//SYSPRINT DD SYSOUT=*"
queue "//SYSUDUMP DD SYSOUT=*"
queue "//SORTOUT  DD SPACE=(CYL,(50,50)),UNIT=SYSDA"
queue "//SYSUT1   DD SPACE=(CYL,(50,50)),UNIT=SYSDA"
queue "//SYSMAP   DD SPACE=(CYL,(50,50)),UNIT=SYSDA"
do i = 1 to tabcnt
  chrcnt = i - 1
  if chrcnt < 10 then chrcnt = "0"chrcnt
  queue "//SYSREC"chrcnt "DD DISP=SHR,DSN=&SYSUID..DB2COPY.TAB"chrcnt
end
queue "//SYSIN    DD DSN=&SYSUID..DB2COPY.SYSPUNCH.MODIFIED,"
queue "//            DISP=SHR"

queue "//************************************************"
queue "//* START FORCE TABLESPACES"
queue "//************************************************"
queue "//STEP0004 EXEC PGM=IKJEFT01,DYNAMNBR=20"
queue "//STEPLIB  DD DSN=DSNP.SDSNLOAD,DISP=SHR"
queue "//SYSPRINT DD SYSOUT=*"
queue "//SYSTSPRT DD SYSOUT=*"
queue "//SYSOUT   DD SYSOUT=*"
queue "//SYSUDUMP DD SYSOUT=*"
queue "//SYSTSIN  DD *"
queue "DSN SYSTEM ("tdss")"

do i = 1 to tabcnt
  queue "-STA DB("jtdb.i") SPACENAM("jtts.i") ACCESS(FORCE)"
end

queue "END"
queue "/*"
```

```
  queue "//************************************************"
  queue "//* DELETE WORK DATASETS"
  queue "//************************************************"
  queue "//STEPØØØ5 EXEC PGM=IEFBR14,COND=(4,LT)"
  queue "//SYSOUT   DD SYSOUT=*"
  queue "//SYSPRINT DD SYSOUT=*"

  do i = 1 to tabcnt
    chrcnt = i - 1
    if chrcnt < 1Ø then chrcnt = "Ø"chrcnt
    queue "//DD"chrcnt "    DD DSN=&SYSUID..DB2COPY.TAB"chrcnt","
    queue "//            DISP=(OLD,DELETE,DELETE)"
  end

  queue "//DDPUNCH  DD DSN=&SYSUID..DB2COPY.SYSPUNCH,"
  queue "//            DISP=(OLD,DELETE,DELETE)"
  queue "//DDPUNCHM DD DSN=&SYSUID..DB2COPY.SYSPUNCH.MODIFIED,"
  queue "//            DISP=(OLD,DELETE,DELETE)"
  queue "//"

  address TSO
  how_many = queued()
  "EXECIO "how_many" DISKW OUTPUT ( FINIS"
  "FREE DD(OUTPUT)"

  address "ISPEXEC"
  "LMINIT DATAID(ID1) DATASET('"jdsn"') ENQ(SHRW)"
  if rc > Ø then
    do
      ZEDSMSG = "LMINIT "||rc
      address "ISPEXEC" "SETMSG MSG(ISRZØØ1)"
      exit rc
    end
  "EDIT DATAID("ID1") MEMBER("jmem")"
  if rc > 4 then
    do
      ZEDSMSG = "EDIT "||rc
      address "ISPEXEC" "SETMSG MSG(ISRZØØ1)"
      exit rc
    end
return;
```

DB2COPYX REXX EXEC

```
/* REXX ***************************************************************/
/*                                                                    */
/* DB2COPYX - MODIFY LOAD CONTROL CARDS                               */
/*            Add REPLACE & ENFORCE NO parms.                         */
/*            Change the CREATOR name.                                */
/*                                                                    */
```

```
/*************************************************************** REXX */
  arg ocreator ncreator
  "alloc ddname(INFILE) shr reuse"
  "alloc ddname(OUTFILE) shrw reuse"
  olen = LENGTH(ocreator)
  dotpos = olen + 7
  "newstack"
   eof_sw = 'n'
   call read_input
   do while eof_sw = 'n'
     if substr(record.1,13,6) = 'LOG NO' then
       do
         tempstr3 = substr(record.1,2Ø,3Ø)
         tempstr2 = 'REPLACE ENFORCE NO '
         tempstr1 = substr(record.1,1,19)
         record.1 = tempstr1||tempstr2||tempstr3
       end
     if substr(record.1,7,olen) = ocreator then
      do
       say substr(record.1,dotpos,1)
       if substr(record.1,dotpos,1) = '.' then
        do
         parse var record.1 tabowner '.' tabname
         record.1 = '     '||ncreator||'.'||tabname
        end
      end
     call write_output
     call read_input
   end
   "execio Ø diskr INFILE (finis"        /* close input file          */
   "execio Ø diskw OUTFILE (finis"       /* close output file         */
   "free ddname(INFILE)"                 /* free input file           */
   "free ddname(OUTFILE)"                /* free output file          */

/*********************************************************************/
/*  read 1 input record                                              */
/*********************************************************************/
read_input:
  "execio 1 diskr INFILE (STEM record."
  if rc <> Ø then eof_sw = 'y'
  return

/*********************************************************************/
/*  write 1 output record                                            */
/*********************************************************************/
write_output:
  "execio 1 diskw OUTFILE (STEM record."
  return
```

Tom Sager (USA) © Xephon 1999

Inserting PL/I declarations into DCLGEN output

Before DB2 Version 4, no indicator variables were generated in the DCLGEN process. In Version 4, DCLGEN was enhanced to generate an array of indicator variables for the host variable structure. However, in my shop the application programmers wanted to have unique indicator host variable names. Furthermore the table owner-id should be removed from the DECLARE TABLE statement of the DCLGEN output, because no owner-id will be specified in host language program EXEC SQL statements. This is necessary for subsystem-independent programs, if the table owner-id differs between test and production. The authorization-id is only determined at bind time via the BIND OWNER option, for matching the owner-id specified in the CREATE TABLE process.

I changed the DCLGEN CLIST DSNEDC01 on library DSN410.SDSNCLST to call a REXX EXEC (DB2DCL), which modifies the DCLGEN output in the desired manner. Passed parameters are table owner-id, table name, and dataset name for the DCLGEN output.

Firstly, the table owner-id is removed from the DECLARE TABLE statement (call OWNREM). Secondly, for each NULL column, an indicator host variable declaration line will be built and queued into the stack (call CREIND). The column name is located using the last four characters (variable TABPRE) of the table name – this is because, in our installation, a table name always has seven characters and the column names have prefixes taken from the last four characters of the table name.

Thirdly, the indicator declarations are read from the stack and inserted into the DCLGEN output. The indicator declarations can either be inserted at the end of all PL/I column DCLs (call ISRINDV1) or after each associated PL/I column DCL (call ISRINDV2). This selection is made by setting the variable VERSION at the beginning of the REXX EXEC.

Performing the DCLGEN function from the DB2I PRIMARY OPTION MENU for a table F#TDB2.WSTEXAM with three NOT NULL and three NULL columns, the following response will be obtained:

```
DSNE944I WARNING, BECAUSE YOUR TABLE OR COLUMN NAMES CONTAIN LOWER CASE
OR NON-ALPHANUMERIC CHARACTERS, YOU WILL NEED TO USE THE SQL (NOT PL/I)
INCLUDE STATEMENT TO INCLUDE DCLGEN OUTPUT INTO YOUR PROGRAM
DSNE903I WARNING, DECLARATION HAS SAME NAME AS TABLE WSTEXAM
DSNE905I EXECUTION COMPLETE, MEMBER WSTEXAM ADDED

DB2DCL :  Table owner id >> F#TDB2 << removed !
DB2DCL :  Indicator variables >> 3 << generated !
```

For Version 1, the DCLGEN output looks like this:

```
   EXEC SQL DECLARE WSTEXAM TABLE
           ( EXAM_FLD01                  CHAR(1) NOT NULL,
             EXAM_FLD02                  CHAR(1),
             EXAM_FLD03                  CHAR(1) NOT NULL,
             EXAM_FLD04                  CHAR(1),
             EXAM_FLD05                  CHAR(1) NOT NULL,
             EXAM_FLD06                  CHAR(1)
           ) ;
/******************************************************************/
/* PLI DECLARATION FOR TABLE F#TDB2.WSTEXAM                       */
/******************************************************************/
 DCL 1 EXAM,
     5 EXAM_FLD01  CHAR(1),
     5 EXAM_FLD02  CHAR(1),
     5 EXAM_FLD03  CHAR(1),
     5 EXAM_FLD04  CHAR(1),
     5 EXAM_FLD05  CHAR(1),
     5 EXAM_FLD06  CHAR(1),
/******************************************************************/
/* THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 6       */
/******************************************************************/
     5 EXAM_FLD02_IND BIN FIXED (15),
     5 EXAM_FLD04_IND BIN FIXED (15),
     5 EXAM_FLD06_IND BIN FIXED (15);
/******************************************************************/
/* THE NUMBER OF INDICATORS DESCRIBED BY THIS DECLARATION IS 3    */
/******************************************************************/
```

And for Version 2, the DCLGEN output looks like this:

```
EXEC SQL DECLARE WSTEXAM TABLE
          ( EXAM_FLD01                    CHAR(1) NOT NULL,
            EXAM_FLD02                    CHAR(1),
            EXAM_FLD03                    CHAR(1) NOT NULL,
            EXAM_FLD04                    CHAR(1),
            EXAM_FLD05                    CHAR(1) NOT NULL,
            EXAM_FLD06                    CHAR(1)
          ) ;
/******************************************************************/
/* PLI DECLARATION FOR TABLE F#TDB2.WSTEXAM                       */
/******************************************************************/
```

```
DCL 1 WSTEXAM,
     5 EXAM_FLDØ1  CHAR(1),
     5 EXAM_FLDØ2  CHAR(1),
     5 EXAM_FLDØ2_IND BIN FIXED (15),
     5 EXAM_FLDØ3  CHAR(1),
     5 EXAM_FLDØ4  CHAR(1),
     5 EXAM_FLDØ4_IND BIN FIXED (15),
     5 EXAM_FLDØ5  CHAR(1),
     5 EXAM_FLDØ6  CHAR(1),
     5 EXAM_FLDØ6_IND BIN FIXED (15);
/*********************************************************************/
/* THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 6          */
/*********************************************************************/
/* THE NUMBER OF INDICATORS DESCRIBED BY THIS DECLARATION IS 3       */
/*********************************************************************/
DSNEDCØ1 CLIST MODIFICATION
```

At the end of the CLIST, after line 238, add the DB2DCL call statement:

```
        %DSNECCØ1                      /* TRANSLATE MSG 294 TOKENS */
        %DB2DCL &DSNEDV21 &DSNEDVØ1 &DSNEDV11
      END                              /* END BIG IF               */
 END                                   /* END WHILE                */
 EXIT
```

REXX EXEC DB2DCL

```
/* REXX */
/*====================================================================-/
/- Function: Edit DB2 DCLGEN output.                                   -/
/-           1. Remove owner-id of table from SQL DCLs.               -/
/-           2. Insert PL/I declaration for indicator variables.      -/
/-              a) Version 1: insert indicator declarations as        -/
/-                            block at end of all PL/I DCLs.          -/
/-              b) Version 2: insert indicator declarations after     -/
/-                            each associated PL/I column DCL.        -/
/-           The procedure is called by the DCLGEN CLIST EXEC         -/
/-           DSNEDCØ1 on library DSN41Ø.SDSNCLST.                     -/
/-           Passed parameters are:                                   -/
/-           1. table owner                                           -/
/-           2. table name                                            -/
/-           3. dataset name for DCLGEN output                        -/
/-====================================================================-*/
arg owner tname dsnin .
ownstr  = '"' || owner || '".'       /* search string for owner-id    */
len     = length(ownstr)
tabpre  = substr(tname,4,4)          /* extract tab column prefix     */
version = '1'                        /* select indicator gen version  */
cre     = 'N'                        /* stop flag for subroutine      */
```

```
cha     = 'N'                          /* stop flag for subroutine      */
qno     =  Ø                           /* number of indicators          */
ADDRESS TSO "ALLOC FI("filein") DA ("dsnin") OLD"
if rc = Ø then
   do
     do forever
        ADDRESS TSO "EXECIO 1 DISKRU "filein" (STEM recin."
        if rc = 2 then leave                              /* eof       */
        call ownrem
        call creind
        if version = '1' then call chaind
     end
     ADDRESS TSO "EXECIO Ø DISKW "filein" (FINIS"
     ADDRESS TSO "FREE  FI("filein")"
     if version = '1' then call isrindv1
     else                  call isrindv2
     say ' '
     say 'DB2DCL :  Table owner id >>' owner '<< removed |'
     say 'DB2DCL :  Indicator variables >>' qno '<< generated |'
     say ' '
   end
exit
/*-------------------------------------------------------------------------- /
/- subroutine: remove table owner-id from SQL DCLs.                         -/
/-------------------------------------------------------------------------- */
ownrem:
pos = pos(ownstr,recin.1)                        /* search for owner-id*/
if pos > Ø then
   do
     str = overlay(' ',recin.1,pos,len)          /* change to blank    */
     str = space(str,1)                          /* single blank-delim */
     recin.1 = ' ' || str                        /* blank trailer      */
                                                 /* rewrite line       */
     ADDRESS TSO "EXECIO 1 DISKW "filein" (STEM recin."
     end
return
/*-------------------------------------------------------------------------- /
/- subroutine:  Create  PL/I declaration for indicator variables.          -/
/-------------------------------------------------------------------------- */
creind:
pos = pos('EXEC SQL',recin.1)                    /* search for tab DCL */
if pos > Ø then
   do
     cre    = 'Y'                               /* begin work on next */
     return                                      /* line               */
     end
if cre  = 'N' then return
/*                                                                        */
pos1 = pos(tabpre,recin.1)                       /* search for column  */
pos2 = pos('NOT NULL',recin.1)                   /* search for NOT NULL*/
```

```
if pos1 > Ø & pos2 = Ø then                      /* is it a NULL col ? */
   do
     pos = pos(tabpre,recin.1)                   /* start of col name  */
     parse var recin.1  =(pos) colname .         /* get col name       */
                                                 /* gen indicator line */
     queue '      5' colname || '_IND' 'BIN FIXED (15),'
   end
if pos(';',recin.1) > Ø then                     /* end of tab DCL ?   */
   do
     cre  = 'N'                                  /* end of work        */
     qno  = queued()                             /* no of indicators   */
   end
return
/*-------------------------------------------------------------------- /
/- subroutine:  Change PL/I DCL statement delimiter; into,            -/
/-              (only required for Version 1)                        -/
/-------------------------------------------------------------------- */
chaind:
pos = pos('DCL 1',recin.1)                       /* search for PL/I DCL*/
if pos > Ø & qno > Ø then
   do
     cha    = 'Y'                                /* begin work on next */
     return                                      /* line               */
     end
if cha  = 'N' then return
/*                                                                    */
if pos(';',recin.1) > Ø then                     /* end of PL/I DCL?   */
   do
      recin.1 = translate(recin.1,',',';')
      ADDRESS TSO "EXECIO 1 DISKW "filein" (STEM recin."
      cha = 'N'                                  /* end of work        */
   end
return
/*-------------------------------------------------------------------- /
/- subroutine:  Insert PL/I declaration for indicator variables       -/
/-              at end of all PL/I column DCLs.                      -/
/-------------------------------------------------------------------- */
isrindv1:
if qno > Ø then
   do
     ADDRESS TSO "ALLOC FI("filein") DA ("dsnin") OLD REUSE"
     newstack
     ADDRESS TSO "EXECIO * DISKR "filein" (FIFO FINIS"
     queue ''                                              /* null line */
     ADDRESS TSO "EXECIO * DISKW "filein
     delstack
     ADDRESS TSO "EXECIO" qno - 1 "DISKW "filein
     pull str                                    /* get last line      */
     pos  = lastpos(',',str)                     /* find line delim    */
     str = overlay(';',str,pos)                  /* change, to;        */
     queue str                                   /* rewrite on stack   */
```

```
     queue  ' /' || left('*',69,'*') || '/'
     queue  ' /* THE NUMBER OF INDICATORS DESCRIBED BY THIS',
         || ' DECLARATION IS' left(qno,3) right('*/',6)
     queue  ' /' || left('*',69,'*') || '/'
     queue ''                                          /* null line */
     ADDRESS TSO "EXECIO * DISKW "filein" (FINIS"
     ADDRESS TSO "FREE  FI("filein")"
   end
return
/*-------------------------------------------------------------------- /
/- subroutine:  Insert PL/I declaration for indicator variables       -/
/-              after each associated PL/I column DCL.               -/
/-------------------------------------------------------------------- */
isrindv2:
if qno > Ø then
   do
     ADDRESS TSO "ALLOC FI("filein") DA ("dsnin") OLD REUSE"
     ADDRESS TSO "EXECIO * DISKR "filein" (STEM recin. FINIS"
/* write lines until  DCL statement                                   */
     do j=1 to recin.Ø
        recout.1 = recin.j
        ADDRESS TSO "EXECIO 1 DISKW "filein" (STEM recout."
        pos = pos('DCL 1',recin.j)           /* search for PL/I DCL  */
        if pos > Ø then  leave
     end
     k = j + 1                               /* pos on next line     */
/* write PLI column and indicator declarations                        */
     do l=1 to qno                           /* loop for queued      */
        pull indstr                          /* indicator lines      */
        parse var indstr . indname .
        do j=k to recin.Ø                    /* is there an indicator*/
           parse var recin.j . colname .     /* line for the column? */
           len2 = length(colname)
           if colname = substr(indname,1,len2) then
              do
                if pos(';',recin.j) > Ø then  /* last PL/I DCL ?     */
                   do
                     recin.j = translate(recin.j,',',';')
                     pos  = lastpos(',',indstr)
                     indstr = overlay(';',indstr,pos) /* change , to ;*/
                   end
                recout.1 = recin.j           /* write column line    */
                ADDRESS TSO "EXECIO 1 DISKW "filein" (STEM recout."
                recout.1 = indstr            /* write indicator line */
                ADDRESS TSO "EXECIO 1 DISKW "filein" (STEM recout."
                leave
              end
           else
              do
                recout.1 = recin.j           /* write column line    */
                ADDRESS TSO "EXECIO 1 DISKW "filein" (STEM recout."
```

```
               end
         end
         k = j + 1
      end
/* write rest of lines                                                */
      do j=k to recin.Ø
         recout.1 = recin.j
         ADDRESS TSO "EXECIO 1 DISKW "filein" (STEM recout."
      end
      recout.1 = ' /* THE NUMBER OF INDICATORS DESCRIBED BY THIS',
          || ' DECLARATION IS' left(qno,3) right('*/',6)
      ADDRESS TSO "EXECIO 1 DISKW "filein" (STEM recout."
      recout.1 = ' /' || left('*',69,'*') || '/'
      ADDRESS TSO "EXECIO 1 DISKW "filein" (STEM recout. FINIS"
      ADDRESS TSO "FREE  FI("filein")"
   end
return
```

Raimund Kleebaur
DBA
Hugo Boss (Germany)

DB2 utility services – part 3

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

DBPARTS

```
)ATTR
 % type(text)    intens(high) skip(on)
 + type(text)    intens(low)  skip(on)
 | type(text)    intens(low)  skip(on) color(yellow)
 [ type(text)    intens(low)  skip(on) color(yellow) hilite(reverse)
 ] type(text)    intens(low)  skip(on) color(red)    hilite(reverse)
 | type(text)    intens(low)  skip(on) color(white)  hilite(reverse)
 _ type(output)  intens(low)  just(right)
 } type(output)  intens(high) just(left) color(red) caps(off)
 $ type(input)   intens(low)
 { type(dataout) intens(high) skip(on) pad(nulls) just(left)
 # type(datain)  intens(high) caps(on) pad(nulls) color(green)
 @ area(dynamic) extend(on)  scroll(on)
)BODY
[ Edit |Include/Exclude Partitions - Table Data         %Row_n1 %of_n2 +
%COMMAND%===>$ZCMD                                    %Scroll ===>$AMT +
```

```
+
+This table holds your included/excluded partitions.
+Enter|G+in command line to continue.
+Set the Incl/Excl flags as you require.
+                        |Incl+
]Database]Tablespace]Part|Excl+
@DATALINE                    @
%
}parmsg                                                          +
|PF3:+Return
)INIT
   &AMT = PAGE
)PROC
   &LVLINE = LVLINE(DATALINE)
   VPUT (LVLINE) SHARED
   VPUT (DATALINE) SHARED
   &CURSOR = .CURSOR
   &CSRPOS = .CSRPOS
)END
```

DBCHECK

```
)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.
//*
//*    CHECK - WAS RUN WITH THE FOLLOWING PARAMETERS:
//*    PARAMETER   PARAMETER VALUE
//*    _____       ___________
//*    SSID     :  &db2
//*    Type     :  &typ
//*    Scope    :  &scop
//*    Creator  :  &creC
//*    Name     :  &tabc
//*    Tsname   :  &tsnc
//*    Dbname   :  &dbnc
//*    Volume   :  &vol
//*    Catname  :  &catn
//*    Tracks   :  &trk
//* ********************************************************************
//*  NUM  DATABASE  TABLESPACE   TRACKS    PART
//*  _    ____      _____         ___      __
```

```
)DOT "ALIST"
//* &detail
)ENDDOT
//*  -  ----  ------  ---   --
//*                       TOTAL:&tot TRACKS OR
//*                             &cyl CYLINDERS
//*--------------------------
//*
)DOT "ALIST"
//*-- TERMINATE CHECK   ---------------
//TRM&scu EXEC PGM=IKJEFT01,COND=(4,LT)
//STEPLIB  DD DSN=DSN410.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
   -TERM UTILITY(&user..CHK&scu)
   END
/*
//*--------------------------
)SEL &pts EQ S
//*-- CHECK ------- &db..&ts
)ENDSEL
)SEL &pts NE S
//*-- CHECK ------- &db..&ts PART &pts
)ENDSEL
//CHK&scu EXEC DSNUPROC,SYSTEM=&db2,
//      UID='&user..CHK&scu',COND=(4,LT)
//STEPLIB DD DSN=DSN410.SDSNLOAD,DISP=SHR
//SORTOUT DD UNIT=3390,VOL=SER=&vol,
)SEL &pts EQ S
//    DSN=&user..CHECK.&ts..SORTOUT.&tsuf,
)ENDSEL
)SEL &pts NE S
//    DSN=&user..CHECK.&ts..P&pts..SORTOUT.&tsuf,
)ENDSEL
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,CATLG),
//    DCB=(BUFNO=20)
//SYSUT1  DD UNIT=3390,VOL=SER=&vol,
)SEL &pts EQ S
//    DSN=&user..CHECK.&ts..SYSUT1.&tsuf,
)ENDSEL
)SEL &pts NE S
//    DSN=&user..CHECK.&ts..P&pts..SYSUT1.&tsuf,
)ENDSEL
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,CATLG),
//    DCB=(BUFNO=20)
//SYSERR  DD UNIT=3390,VOL=SER=&vol,
)SEL &pts EQ S
```

```
//    DSN=&user..CHECK.&ts..SYSERR.&tsuf,
)ENDSEL
)SEL &pts NE S
//    DSN=&user..CHECK.&ts..P&pts..SYSERR.&tsuf,
)ENDSEL
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,CATLG),
//    DCB=(BUFNO=2Ø)
//SYSIN  DD *
)SEL &typ = BOTH | &typ = DATA
)SEL &pts EQ S
 CHECK  DATA TABLESPACE &db..&ts
)ENDSEL
)SEL &pts NE S
 CHECK  DATA TABLESPACE &db..&ts PART &pts
)ENDSEL
       SCOPE &scop
    SORTDEVT 339Ø
)ENDSEL
)SEL &typ = BOTH | &typ = INDEX
 CHECK INDEX TABLESPACE &db..&ts
    SORTDEVT 339Ø
)ENDSEL
)ENDDOT
/*
```

DBCOPY

```
)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.
//*
//*    COPY - WAS RUN WITH THE FOLLOWING PARAMETERS:
//*    PARAMETER   PARAMETER VALUE
//*    _____       ___________
//*    SSID     :  &db2
//*    Creator  :  &creC
//*    Name     :  &tabc
//*    Tsname   :  &tsnc
//*    Dbname   :  &dbnc
//*    Card     :  &card
```

```
//*    Full     :  &ful
//*    Shrlevel :  &ref
//*    Copypref :  &pref
//*    Stopts   :  &sto
//*    Quiesce  :  &que
//*    Offsite  :  &off
//*    Devt     :  &dev
//*    Volume   :  &vol
//*    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=IKJEFT01,COND=(4,LT)
//STEPLIB  DD DSN=DSN410.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(UT)
)ENDSEL
)SEL &pts NE S
   -START DATABASE(&db) SPACENAM(&ts) PART(&pts) ACCESS(UT)
)ENDSEL
)ENDDOT
//*
)ENDSEL
)SEL &que EQ YES AND &varx EQ NE
//*-- TERMINATE UTILITY ---------------
//TERMQU  EXEC PGM=IKJEFT01,COND=(4,LT)
//STEPLIB  DD DSN=DSN410.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
   -TERM UTILITY(&user..QUIEUT)
/*
//*-----------------------------------------------
//*-- QUIESCE STEP                          ----
```

```
//QUIEUT EXEC DSNUPROC,SYSTEM=&db2,
//      UID='&user..QUIEUT',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
//*── TERMINATE UTILITY ────────────────
//TERMCO EXEC PGM=IKJEFTØ1,COND=(4,LT)
//STEPLIB  DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
)SEL &varx = JA
   -TERM UTILITY(&user..COPYX)
)ENDSEL
   -TERM UTILITY(&user..COPY)
/*
)SEL &varx = JA
//*── COPY  DSNDBØ1.SYSUTILX ────────────
//COPYX EXEC DSNUPROC,SYSTEM=&db2,
//      UID='&user..COPYX',UTPROC='',COND=(4,LT)
//STEPLIB DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
)DOT "ALIST"
)SEL &db EQ DSNDBØ1 AND &ts EQ SYSUTILX
//COPY1  DD UNIT=&dev,VOL=SER=&vol,
//    DSN=&pref..&db..&ts..&dsufb,
)SEL &dev = 339Ø
//    DCB=(BUFNO=2Ø,BLKSIZE=22528),
//    DISP=(,CATLG,DELETE),
//    SPACE=(TRK,(&pri,&sec,),RLSE)
)ENDSEL
)SEL &dev = TAPE
//    DISP=(NEW,KEEP,KEEP),
//    LABEL=(&scu,SL)
)ENDSEL
)SEL &off = YES
//RDDN1  DD UNIT=&dev,VOL=SER=&vol,
//    DSN=&pref..R.&db..&ts..&dsufb,
)SEL &dev = 339Ø
//    DCB=(BUFNO=2Ø,BLKSIZE=22528),
//    DISP=(NEW,CATLG,CATLG),
```

```
//    SPACE=(TRK,(&pri,&sec,),RLSE)
)ENDSEL
)SEL &dev = TAPE
//    DISP=(NEW,KEEP,KEEP),
//    LABEL=(1,SL)
)ENDSEL
)ENDSEL
//SYSIN  DD  *
)BLANK 1
  COPY  TABLESPACE &db..&ts
           COPYDDN COPY1
)SEL &off = YES
       RECOVERYDDN RDDN1
)ENDSEL
              FULL &ful
          SHRLEVEL CHANGE
)ENDSEL
)ENDDOT
)BLANK 1
)ENDSEL
//*── COPY ──────────────────
//COPYS EXEC DSNUPROC,SYSTEM=&db2,
//      UID='&user..COPY',UTPROC='',COND=(4,LT)
//STEPLIB DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
)DOT "ALIST"
)SEL &ts NE SYSUTILX
//COPY&scu DD UNIT=&dev,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
)SEL &dev = 339Ø
//    DCB=(BUFNO=2Ø,BLKSIZE=22528),
//    DISP=(,CATLG,DELETE),
//    SPACE=(TRK,(&pri,&sec,),RLSE)
)ENDSEL
)SEL &dev = TAPE
//    DISP=(NEW,KEEP,KEEP),
//    LABEL=(&scu,SL)
)ENDSEL
)SEL &off = YES
//RDDN&scu DD UNIT=&dev,VOL=SER=&vol,
)SEL &pts EQ S
//    DSN=&pref..R.&db..&ts..&dsufb,
)ENDSEL
)SEL &pts NE S
//    DSN=&pref..R.&db..&ts..P&pts..&dsufb,
)ENDSEL
```

```
)SEL &dev = 339Ø
//    DCB=(BUFNO=2Ø,BLKSIZE=22528),
//    DISP=(NEW,CATLG,CATLG),
//    SPACE=(TRK,(&pri,&sec,),RLSE)
)ENDSEL
)SEL &dev = TAPE
//    DISP=(NEW,KEEP,KEEP),
//    LABEL=(&scu,SL)
)ENDSEL
)ENDSEL
)ENDSEL
)ENDDOT
//SYSIN  DD  *
)DOT "ALIST"
)SEL &ts NE SYSUTILX
)BLANK 1
  COPY  TABLESPACE &db..&ts
)SEL &pts NE S
             DSNUM &pts
)ENDSEL
           COPYDDN COPY&scu
)SEL &off = YES
       RECOVERYDDN RDDN&scu
)ENDSEL
              FULL &ful
          SHRLEVEL &ref
)ENDSEL
)ENDDOT
/*
)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
```

DBMODI

```
)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
//*
//*    MODIFY - WAS RUN WITH THE FOLLOWING PARAMETERS:
//*    PARAMETER   PARAMETER VALUE
//*    ─────       ─────────────
//*    SSID     :  &db2
//*    Delage   :  &dage
//*    Deldate  :  &ddate
//*    Tsname   :  &tsnc
//*    Dbname   :  &dbnc
//* ****************************************************************
//* NUM DATABASE TSNAME   ICDATE     DSNAME
//* -   ────     ────     ─────      ──────────────────
)DOT "MLIST"
//* &detail
)ENDDOT
//* - ──── ──── ─── ──────────────────
//*
//*── MODIFY - TSO DELETE DATASET(S) ────────
//DELTSO EXEC PGM=IKJEFT01
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
)DOT "MLIST"
 DELETE '&dsn'
)ENDDOT
//*── MODIFY RECOVERY ─────────────────
//MODIFY EXEC DSNUPROC,SYSTEM=&db2,
//      UID='&user..MODIFY',UTPROC=''
//STEPLIB  DD DSN=DSN410.SDSNLOAD,DISP=SHR
//SYSIN    DD  *
)DOT "DLIST"
)SEL &dage GE 0
MODIFY RECOVERY TABLESPACE &DB..&TS DELETE AGE &dage
)ENDSEL
)SEL &ddate > 0
MODIFY RECOVERY TABLESPACE &DB..&TS DELETE DATE &ddated
)ENDSEL
)ENDDOT
/*
```

DBQUIE

```
)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
//*
//*    QUIESCE - WAS RUN WITH THE FOLLOWING PARAMETERS:
//*    PARAMETER   PARAMETER VALUE
//*    ─────       ─────────
//*    SSID     :  &db2
//*    Creator  :  &creC
//*    Name     :  &tabc
//*    Tsname   :  &tsnc
//*    Dbname   :  &dbnc
//* ****************************************************************
//*  NUM  DATABASE  TABLESPACE
//*  -    ────      ─────
)DOT "ALIST"
//* &detail
)ENDDOT
//*  -  ────  ─────
//*──────────────────────
//*
//*── TERMINATE UTILITY ──────────────
//TERMQU  EXEC PGM=IKJEFTØ1,COND=(4,LT)
//STEPLIB  DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
   -TERM UTILITY(&user..QUIEUT)
/*
//*──────────────────────────────────────────────
//*── QUIESCE STEP                          ────
//QUIEUT EXEC DSNUPROC,SYSTEM=&db2,
//      UID='&user..QUIEUT',UTPROC='',COND=(4,LT)
//STEPLIB  DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
//SYSIN    DD  *
  QUIESCE
)DOT "ALIST"
     TABLESPACE &DB..&TS
)ENDDOT
/*
```

DBRECO

```
)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.
//*
//*    RECOVERY - WAS RUN WITH THE FOLLOWING PARAMETERS:
//*    PARAMETER   PARAMETER VALUE
//*    ─────       ─────────────
//*    SSID     :  &db2
//*    Creator  :  &crec
//*    Name     :  &tabc
//*    Tsname   :  &tsnc
//*    Dbname   :  &dbnc
//*    Rectype  :  &rtyp
//*    Datefrom :  &dfr
//*    Dateto   :  &dto
//*    Timefrom :  &tfr
//*    Timeto   :  &tto
//*    Volume   :  &vol
//*    Volsort  :  &vola
//*    Volsort  :  &volb
//*    Catname  :  &catn
//*    Tracks   :  &trk
//* ****************************************************************
//*  NUM  DATABASE  TABLESPACE   TRACKS   PART
//*  -    ────      ─────        ───      ──
)DOT "ALIST"
//* &detail
)ENDDOT
//*  -  ────  ─────  ───   ──
//*                      TOTAL:&tot TRACKS OR
//*                            &cyl CYLINDERS
//*──────────────────────────
//*
)DOT "ALIST"
//*── TERMINATE UTILITY ──────────────
//TREC&scu  EXEC PGM=IKJEFT01,COND=(4,LT)
//STEPLIB  DD DSN=DSN410.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
```

```
   -TERM UTILITY(&user..REC&scu)
/*
//*-----------------------------
)SEL &pts EQ S
//*-- RECOVER ------ &db..&ts
)ENDSEL
)SEL &pts NE S
//*-- RECOVER ------ &db..&ts PART &pts
)ENDSEL
//RECO&scu EXEC DSNUPROC,SYSTEM=&db2,
//      UID='&user..REC&scu',UTPROC=''
//STEPLIB DD DSN=DSN41Ø.SDSNLOAD,DISP=SHR
//SORTWKØ1 DD UNIT=339Ø,VOL=SER=&vola,
//    DSN=&&&&SORTWK1,
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,DELETE)
//SORTWKØ2 DD UNIT=339Ø,VOL=SER=&volb,
//    DSN=&&&&SORTWK2,
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,DELETE)
//SORTWKØ3 DD UNIT=339Ø,VOL=SER=&vola,
//    DSN=&&&&SORTWK3,
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,DELETE)
//SORTWKØ4 DD UNIT=339Ø,VOL=SER=&volb,
//    DSN=&&&&SORTWK4,
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,DELETE)
//SORTOUT DD UNIT=339Ø,VOL=SER=&vol,
)SEL &pts EQ S
//    DSN=&user..&db..&ts..OUT.&tsuf,
)ENDSEL
)SEL &pts NE S
//    DSN=&user..&db..&ts..P&pts..OUT.&tsuf,
)ENDSEL
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,CATLG)
//SYSREC  DD UNIT=339Ø,VOL=SER=&vol,
)SEL &pts EQ S
//    DSN=&user..&db..&ts..REC.&tsuf,
)ENDSEL
)SEL &pts NE S
//    DSN=&user..&db..&ts..P&pts..REC.&tsuf,
)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.&tsuf,
)ENDSEL
```

```
)SEL &pts NE S
//    DSN=&user..&db..&ts..P&pts..UT1.&tsuf,
)ENDSEL
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,CATLG)
//SYSIN  DD *
)BLANK 1
 RECOVER TABLESPACE &db..&ts
)SEL &pts NE S
              DSNUM &pts
)ENDSEL
)SEL &rtyp = RBA
              TORBA X'&rbadsn'
)ENDSEL
)SEL &rtyp = TOCOPY
             TOCOPY &rbadsn
           TOVOLUME CATALOG
)ENDSEL
)BLANK 1
 RECOVER INDEX (ALL)
         TABLESPACE &db..&ts
)BLANK 1
)ENDDOT
/*
```

DBREPO

```
)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
//*
//*    REPORT - WAS RUN WITH THE FOLLOWING PARAMETERS:
//*    PARAMETER   PARAMETER VALUE
//*    _____       ___________
//*    SSID     :  &db2
//*    Creator  :  &creC
//*    Name     :  &tabc
//*    Tsname   :  &tsnc
//*    Dbname   :  &dbnc
//*    Type     :  &rtype
//*    Current  :  &rcu
//*    Summary  :  &rsu
//* ********************************************************************
```

```
//*  NUM  DATABASE  TABLESPACE
//*  -    ----      ------
)DOT "ALIST"
//* &detail
)ENDDOT
//*  -  -----  ------
//*-------------------------
//*
//*-- TERMINATE UTILITY ----------------
//TERMRP  EXEC PGM=IKJEFT01,COND=(4,LT)
//STEPLIB  DD DSN=DSN410.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
   -TERM UTILITY(&user..REPORT)
/*
//*---------------------------------------
//*-- REPORT            ----------------
//REPSTP EXEC DSNUPROC,SYSTEM=&db2,
//      UID='&user..REPORT',UTPROC=''
//STEPLIB  DD DSN=DSN410.SDSNLOAD,DISP=SHR
//SYSIN    DD  *
)DOT "ALIST"
)SEL &rtype = RECOVERY | &rtype = BOTH
  REPORT RECOVERY TABLESPACE &db..&TS
         DSNUM ALL
)SEL &rcu = YES
         CURRENT
)ENDSEL
)SEL &rsu = YES
         SUMMARY
)ENDSEL
)ENDSEL
)SEL &rtype = TABLESPACESET | &rtype = BOTH
  REPORT TABLESPACESET TABLESPACE &db..&TS
)ENDSEL
)ENDDOT
/*
```

DBRUNS

```
)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
//*
//*    RUNSTATS - WAS RUN WITH THE FOLLOWING PARAMETERS:
//*    PARAMETER   PARAMETER VALUE
//*    _____       ____________
//*    SSID     :  &db2
//*    Creator  :  &creC
//*    Name     :  &tabc
//*    Tsname   :  &tsnc
//*    Dbname   :  &dbnc
//*    Shrlevel :  &ref
//*    Report   :  &rre
//*    Update   :  &upd
//* ********************************************************************
//*  NUM  DATABASE  TABLESPACE
//*  _    ____      _____
)DOT "ALIST"
//* &detail
)ENDDOT
//*  _  ____  _____
//*-------------------------
//*
//*-- TERMINATE UTILITY ----------------
//TERMRU  EXEC PGM=IKJEFT01,COND=(4,LT)
//STEPLIB  DD DSN=DSN410.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
   -TERM UTILITY(&user..RUNSTAT)
/*
//*---------------------------------------
//*-- RUNSTATS          ----------------
//REPSTP EXEC DSNUPROC,SYSTEM=&db2,COND=(4,LT),
//      UID='&user..RUNSTAT',UTPROC=''
//STEPLIB  DD DSN=DSN410.SDSNLOAD,DISP=SHR
//SYSIN    DD  *
)DOT "ALIST"
)BLANK 1
  RUNSTATS TABLESPACE &db..&TS
               TABLE (ALL)
               INDEX (ALL)
             SHRLEVEL &ref
               REPORT &rre
               UPDATE &upd
)ENDDOT
/*
```

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

Bernard Zver
Database Administrator
Informatika Maribor (Slovenia)

Year 2000 and select current date

By now most sites will be active with their Year 2000 testing phases, and will probably be using date shifting tools for at least some of their testing. These applications normally work by intercepting SVC 11.

At our site we use a home-grown application rather than a purchased option and, because this was published in *MVS Update*, Issue 144, September 1998, it is possible that other sites are also using this application.

Whether you use a purchased option or not, any application which solely works on SVC11 will be subject to the problem we recently encountered with DB2 and SVC11 interceptions. It wasn't a failure, it just turned out that, if DB2 was asked to select the current date, it did just that (ie the date request was not being intercepted). Therefore, DB2 was using an STCK for this function rather than the SVC.

This article documents a fix that allows DB2 to have its date altered. Unfortunately, it's not quite as simple as replacing the STCK with the SVC request because, during the diagnosing of this problem, it was found that DB2 is not in a state that supports SVC operation at the time it is requesting the date.

I have created a ZAP for module DSNXGRDS that allows DB2 to be modified such that it will operate an offset number of days into the future. This fix works for DB2 Version 4, and, although Version 5 has not been installed yet, I do know the offsets are unchanged and should therefore be OK. However, if you wish to try this ZAP, please double-check it on a test system first (and that includes your Version 4 DB2).

In essence, the ZAP redirects the STCK back to some code I have developed which has been inserted into the CSECT DSNXVCTS over the module eye-catcher at the start. This then adds the days onto the STCK retrieved information, and branches back after the STCK.

Please note that by doing this I have prevented DB2 from coping with a situation involving a problem clock. However, because this is a very unlikely situation, I doubt whether this will be a problem.

If the 'days to add' field is left as zero, DB2 will operate as normal (see ZAP below). To modify the date, either change this field via a storage alter (eg using SYSVIEW/E, OMEGAMON, etc) or ZAP the module before it is loaded.

Ultimately my aim is to make this system more dynamic, but, given the rapidity with which the millennium is approaching, it may be more useful to have this ZAP in its present form than wait for the full application.

```
//ZAPSELC JOB your job card
//A       EXEC PGM=AMASPZAP
//SYSPRINT DD  SYSOUT=*
//SYSLIB   DD  DISP=SHR,DSN=DB2.load.library
//SYSIN    DD  *
 *
  NAME  DSNXGRDS DSNXVCTS
  VER   0004     10C4E2D5
  VER   0028     C4E2D5E7
  VER   00EA     B205D098,4780C108
  REP   0004     580D0098
  REP   0008     5A0C0028
  REP   000C     500D0098
  REP   0010     47FC00F2
 *
 * the offset below (0028) contains the days offset to add
 * calculated as follows:
 * (days_difference*24*60*60)/1.048576 all converted to hex
 *
  REP  0028    00000000
  REP  00EE    47FC0004
  REP  00F2    47F0C108
/*
The code that has been inserted translates as follows:
L    R0,152(R13)     * at offset 0004
A    R0,40(12)       * at offset 0008
ST   R0,152(R13)     * at offset 000C
B    242(R12)        * at offset 0010

B    4(R12)          * at offset 00EE
B    264(R12)        * at offset 00F2
```

Bluestone Software has released Sapphire/Web for DB2 UDB, a repository and application server framework that extends Web deployment opportunities. Sapphire/Web application server now runs on AS/400 servers, providing a means for the development, deployment, integration, and management of enterprise applications on IBM's mid-range machines.

In addition to AS/400, Sapphire/Web runs on AIX, OS/390, Windows 95/98 and NT, Linux, and other versions of Unix. Bluestone also supports deployment options including Java, JavaBeans, Enterprise JavaBeans, C, C++, and CORBA services.

With Sapphire/Web for DB2 Universal Database, users get native support for DB2 and can develop new applications that exploit Web technologies as well as DB2, CICS, and MQSeries applications.

For further information contact:
Bluestone Software, 1000 Briggs Road, Mount Laurel, NJ 08054, USA.
Tel: (609) 727 4600.
URL: http://www.bluestone.com.

* * *

DB2 users can benefit from Version 4.1 of Neon Systems' ShadowDirect integration middleware for System/390. This incorporates DB2, IMS/DB, IMS/TM, CICS, ADABAS, VSAM, and all other sources into ODBC, application server, and common development tool execution environments.

Version 4.1 includes added support for IBM's Work Load Manager, DB2 stored procedure access, dynamic load-balancing, ADABAS access, support for Microsoft Transaction Server, as well as access to OS/390 and MVS for Forte and BEA Tuxedo/M3 users. Also new is integration with automated operations software packages through a new command rule capability.

For further information contact:
Neon Systems, 14141 Southwest Freeway, Suite 6200, Sugar Land, TX 77478, USA.
Tel: (281) 491 4200.
URL: http://www.neonsys.com.

* * *

IBM has announced DataPropagator Relational 5.1, providing an efficient way of automatically maintaining consistent copies of relational data in the DB2 family of databases. Version 5.1 enhancements include easier intuitive administration from OS/2 and Windows; subscription sets for transaction consistency; update-anywhere and on-demand replication; and support for remote journalling.

For further information contact your local IBM representative.