101 DB2

March 2001

update

In this issue

DB2 Update

Published by

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

North American office

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

Subscriptions and back-issues

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

DB2 Update on-line

Code from *DB2 Update* can be downloaded from our Web site at http://www.xephon.com/db2update.html; you will need the user-id shown on your address label.

Editor

Trevor Eddolls

Disclaimer

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

Contributions

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

Recover an accidentally dropped table

WHAT HAPPENED

There was one tablespace with one table, and that table was defined with x columns. Afterwards we altered it to contain x+y columns, but no REORG was done. Back-ups were taken using full image copies. We dropped the table, but the tablespace still exists.

What could be done to restore this table? What follows is a test scenario to replay the events.

In summary:

- Create the tablespace and table to play with. Get the information and fill the table.
- Take a full image copy.
- Alter the table and insert some more values.
- Drop the table.
- Prepare for a restore – try to get the RBA of the drop in the logs.
- Recover the tablespace to the RBA of drop -1 – so you don't lose any updates after the last imagecopy.
- DSN1COPY the tablespace to a newly-created VSAM file.
- Recreate the original table.
- DSN1COPY back from the newly-created VSAM file to the tablespace with OBITXLAT.
- Redo the 'alter' table.
- Do a select on the table.
- Prohibit drops in the future.

In more detail, the code looks like this:

```
 --CREATE TABLESPACE TO PLAY WITH
 -- DSN=DB2T.DSNDBC.BROL.BROL1.I0001.A001
 CREATE TABLESPACE BROL1
          IN BROL
          USING STOGROUP SGTPAR00
            PRIQTY  7000
            SECQTY  1000
            ERASE YES
          FREEPAGE 0
          PCTFREE  5
          LOCKSIZE ANY
          BUFFERPOOL BP0
          CLOSE NO;

 --CREATE TABLE      TO PLAY WITH
 CREATE TABLE  TSHVR.BROL1
 (
   FIELD1              CHAR(1)           NOT NULL WITH DEFAULT ,
   FIELD2              CHAR(4)           NOT NULL WITH DEFAULT ,
   FIELD3              CHAR(4)           NOT NULL WITH DEFAULT
 )
 IN  BROL.BROL1;

--GET DBID PSID TO CHECK AFTERWARDS
   SELECT
    NAME,DBNAME,DBID,PSID,NTABLES
    FROM SYSIBM.SYSTABLESPACE
    WHERE NAME='BROL1';
!!OUTPUT:DBID=272 PSID=2
--GET OBID      TO CHECK AFTERWARDS
 SELECT
   NAME,CREATOR,DBNAME,TSNAME,DBID,OBID,COLCOUNT
  FROM SYSIBM.SYSTABLES
  WHERE NAME='BROL1' AND CREATOR='TSHVR';
!!OUTPUT:DBID=272 OBID=3

 INSERT INTO TSHVR.BROL1
  VALUES('1','1AAA','1BBB');
 INSERT INTO TSHVR.BROL1
  VALUES('2','2AAA','2BBB');

 -STOP    DATABASE(BROL) SPACENAM(BROL1)
   (THIS STOP ONLY TO MAKE SURE THAT ALL PAGES ARE WRITTEN OUT
    FOR NEXT DSN1PRNT)

 //*CHECK TO SEE THAT DBID,PSID,OBID ARE THE SAME VALUES AS FROM SELECT
 //RUNPRNT EXEC PGM=DSN1PRNT,PARM='FORMAT,PRINT'
 //SYSPRINT DD SYSOUT=X
```

```
 //SYSUT1   DD DISP=SHR,DSN=DB2T.DSNDBC.BROL.BROL1.I0001.A001
 !!OUTPUT: HPGOBID='01100002'X   (dbid 0X0110=272 psid 0x0002=2)
            PGSOBD='0003'X        (obid 0x0003=3)

 -START   DATABASE(BROL) SPACENAM(BROL1)

//*TAKE FULL IMAGE COPY
//DBB00001 EXEC PGM=DSNUTILB,PARM=DSNT,
// REGION=4096K
//D000001A DD DISP=(NEW,CATLG),
// STORCLAS=DEFAULT,SPACE=(CYL,(1,1)),
// DSN=TSHVR.DSNT.BROL.BROL1.COPY#
//SYSPRINT DD SYSOUT=X
//SYSIN DD *
 COPY TABLESPACE BROL.BROL1 COPYDDN(D000001A)
 FULL YES SHRLEVEL REFERENCE
/*

 --ALTER TABLE AND INSERT SOME MORE VALUES
 ALTER TABLE TSHVR.BROL1
  ADD FIELD4    CHAR(4);

 INSERT INTO TSHVR.BROL1
  VALUES('4','4AAA','4BBB','4CCC');

 --DROP IT!!!!
 DROP  TABLE TSHVR.BROL1;

 --PREPARE RESTORE BEGIN
 RUN COMMAND -ARCHIVE LOG
  TO OFFLOAD CURRENTLY ACTIVE LOG TO ARCHIVE LOG DATASETS

 RUN UTILITY REPORT RECOVERY
  TO FIND DATE,TIME,RBA OF DROP :
 REPORT RECOVERY TABLESPACE BROL.BROL1
 !!OUTPUT:
  102400    14053815   00036ACC4BAE   00036ACC7E37
  102400    14590861   00036CCDA172   000000000000

 RUN UTILITY DSNJU004
    TO KNOW WHICH ARCHIVE LOG DATASET CONTAINS RBA-RANGE
    AROUND TIME OF DROP
 !!OUTPUT:
  00036B014000 00036CCE4FFF IN DSN410C.AL2.D00298.T1510423.A0000418

 RUN DSN1LOGP ON ARCHIVE LOG WHICH CONTAINS TIME OF DROP
  TRY TO FIND LAST RBA BEFORE DROP
  LOGPOINT WITH MANY SYSTEM TABLES IS PROBABLY RBA OF DROP
```

```
//STEP1 EXEC PGM=DSN1LOGP
//SYSPRINT DD SYSOUT=X
//ARCHIVE   DD DISP=SHR,DSN=DSN410C.AL2.D00298.T1510423.A0000418
//SYSSUMRY DD SYSOUT=X
//SYSIN    DD *
   SUMMARY(YES)
   DATAONLY(YES)
   RBASTART(00036B014000)   RBAEND(00036CCE4FFF)
   DBID (0110) OBID(0003)
/*
 !!OUTPUT:
 DSN1LPRT  UR  CONNID=BATCH     CORRID=TSHVRA        AUTHI
    START DATE=00.298 TIME=15:08:17  DISP=COMMITTED   INF
    STARTRBA=00036CCDE837  ENDRBA=00036CCE2171  STARTLRSN
    LUWID=KVVCR.DB2LU.B4D74D31BEF4.0001  COORDINATOR=*
    PARTICIPANTS=*
    DATA MODIFIED:
         DATABASE=0110=BROL      PAGE SET=0002=BROL1
         DATABASE=0006=DSNDB06   PAGE SET=0074=DSNDCX01
         DATABASE=0006=DSNDB06   PAGE SET=0009=SYSDBASE
         DATABASE=0006=DSNDB06   PAGE SET=0061=DSNATX01
         DATABASE=0006=DSNDB06   PAGE SET=0062=DSNATX02
         DATABASE=0006=DSNDB06   PAGE SET=00D3=DSNATX03
         DATABASE=0006=DSNDB06   PAGE SET=005D=DSNDTX01
         DATABASE=0006=DSNDB06   PAGE SET=00A4=DSNDTX02
         DATABASE=0001=DSNDB01   PAGE SET=001F=DBD01

-- CHECK TABLESPACE CONTAINS ONLY 1 TABLE.so you don't destroy other
tables
 SELECT A.NAME,A.DBNAME,A.PARTITIONS,A.NTABLES
  FROM SYSIBM.SYSTABLESPACE AS A
  WHERE A.NAME='BROL1' AND A.DBNAME='BROL' AND
   A.NTABLES<2 ;
//RECOVER EXEC DSNUPROC
//SYSPRINT DD SYSOUT=*
//SYSIN    DD  *
  RECOVER TABLESPACE BROL.BROL1
   TORBA X'00036CCDE836'
/*

 -STOP    DATABASE(BROL) SPACENAM(BROL1)

 DEFINE CLUSTER(NAME('TSHVR.BROL.BROL1.VSAM.C001#')
  MODEL('DB2T.DSNDBC.BROL.BROL1.I0001.A001'))

//RUNCOPY EXEC PGM=DSN1COPY,PARM='FULLCOPY,CHECK'
//SYSPRINT DD SYSOUT=X
//SYSUT1   DD DISP=OLD,DSN=DB2T.DSNDBC.BROL.BROL1.I0001.A001
```

```
//SYSUT2    DD DISP=OLD,DSN=TSHVR.BROL.BROL1.VSAM.C001#
//*

 -START   DATABASE(BROL) SPACENAM(BROL1)

--RECREATE THE ORIGINAL TABLE
 CREATE TABLE  TSHVR.BROL1
 (
   FIELD1              CHAR(1)           NOT NULL WITH DEFAULT ,
   FIELD2              CHAR(4)           NOT NULL WITH DEFAULT ,
   FIELD3              CHAR(4)           NOT NULL WITH DEFAULT
 )
 IN  BROL.BROL1;

 SELECT
   NAME,CREATOR,DBNAME,TSNAME,DBID,OBID,COLCOUNT
  FROM SYSIBM.SYSTABLES
  WHERE NAME='BROL1' AND CREATOR='TSHVR';
  !!OUTPUT:DBID=272 OBID=4

 -STOP    DATABASE(BROL) SPACENAM(BROL1)

//RUNCOPY EXEC PGM=DSN1COPY,PARM='OBIDXLAT,RESET,FULLCOPY,CHECK'
00001513
//SYSPRINT DD SYSOUT=X
//SYSUT1    DD DISP=OLD,DSN=TSHVR.BROL.BROL1.VSAM.C001#
//SYSUT2   DD DISP=OLD,DSN=DB2T.DSNDBC.BROL.BROL1.I0001.A001
//SYSXLAT   DD *
272,272
 2,2
 3,4
/*
//*

 -START   DATABASE(BROL) SPACENAM(BROL1)

 ALTER TABLE TSHVR.BROL1
  ADD FIELD4    CHAR(4);

 SELECT * FROM TSHVR.BROL1;
   All rows and columns should be there again!

 TO AVOID THIS KIND OF PROBLEMS:
  ALTER TABLE TSHVR.BROL1 DROP  RESTRICT ON DROP;
```

Herman Vierendeels
Systems Programmer (Belgium)

Are you recursive?

A fuzzy SELECT (see *Fuzzy Select, DB2 Update*, Issue 87, January 2000 for a full description) may require recursion to satisfy some fuzzy queries. Recursion in DB2 means referring to the same table as in a *recursive join.* Consider the following logical table:

```
COURSE(COURSE_ID, SCHOOL_ID, course_name, course_duration, min_classize,
max_classize, COURSE_ID_prerequisite)
```

where: COURSE_ID is the primary key; SCHOOL_ID is the foreign key to the school that 'owns' the course; and COURSE_ID_prerequisite is a recursive foreign key to the course table. A prerequisite is a qualifier (DB2 and ANSI relational databases do not allow the same column name within the same table). For simplicity, it is assumed that each course has zero or one prerequisite.

To get prerequisite course information requires joining the course table to itself using the recursive foreign key:

```
SELECT      c.course_id_prerequisite, p.course_name, p.school_id,
p.course_name, p.course_duration, p.min_classize, p.max_classize,
p.course_id_prerequisite
      --c. & p. are column qualifiers
FROM            course c, course p
      --c & p are table designators
      --listing a course twice in FROM makes it a recursive join
WHERE      p.course_id = c.course_id_prerequisite
```

The result table from the above SELECT returns the specified column values for the prerequisite course. The prerequisite course can have a prerequisite course. To get all the prerequisite courses requires a *Bill of Material* explosion discussed below.

Logic University has a student counselling program where juniors and seniors advise freshmen and sophomores, and graduate students counsel juniors and seniors. The logical table is:

```
STUDENT(STUDENT_ID, student_name, STUDENT_ID_advisor, …)
```

A recursive join similar to course (above) will retrieve the advisor or NULL. Course:prerequisite is a 1:1 relationship as specified in the recursive foreign key above; student:advisor is a 1:M relationship,

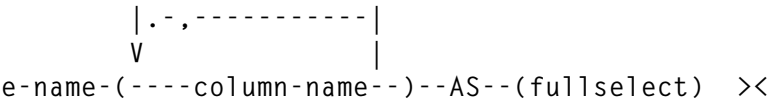
where a student can have zero or one advisor, but an advisor can have zero, one, or many students.

RECURSION (ALIAS BILL OF MATERIAL)

BOM explosions using SQL require a recursive common table expression. Here are the definitions:

- A table expression creates a temporary result table from a simple query.
- A common table expression defines a temporary result table with a table name specifiable in any FROM clause of the following fullselect or subsequent common table expressions.
- A recursive common table expression is created by having a fullselect of a common table expression reference itself in a FROM clause.

An application program could do an explosion without a recursive common table expression. This requires starting a new query for each recursion level and probably storing the results in the database for ordering (application programs written in COBOL could use the SORT verb). SQL using a recursive common table expression is complex, but less so than an application program. The recursive common table expression syntax is:



A recursive common table expression requires a qualified column name. The column name list must contain as many names as the fullselect result table.

The following SQL is for a query that asks what all the prerequisite courses for course x are:

```
WITH preq_course (course_id, course_name, …) AS
--preq_course is common table expression
       (SELECT      c.course_id, c.course_id_prerequisite
--with select is initiation fullselect; it references source table;
  --it cannot reference preq_course
       FROM            course c
```

```
            WHERE      c.course_id = 401    --401 is specified course_id
            UNION ALL                       --all is required
            SELECT     p.course_id, p.course_id_prerequisite
            FROM       preq_course      parent, course child
            --preq_course is recursive common table expression
            WHERE      parent.course_id_prerequisite = child.course_id)
            SELECT DISTINCT      course_id_prerequisite
--distinct not needed for this query since it is 1:1 but is for any
--M relationship (1:M, M:1, M:M)
          FROM preq_course      --gets row from temporary result table;
                                --works like correlated subquery
            ORDER BY   course_id      DESC
```

Assume a hierarchical course_id where any course with a leading digit of 1 is a freshman course, 2 is a sophomore, 3 is a junior, and 4 is a senior, and last two digits are the same as for a course package such as English 101, English 201, etc. Then the result table would be:

```
401 is    401  301  201  101
301 is         301  201  101
201 is              201  101
101 is                   101
```

The query above is a single-level explosion. Assume you want to know how many students a student advisor is advising. This requires a summarized explosion:

```
WITH student_advised (student_id, student_name, student_id_advisor, …)
AS
          (SELECT     s.student_id, s.student_id_advisor
          FROM            student s
          WHERE       s.student_id_advisor = 169
--169 is the specified student_id_advisor
          UNION ALL
          SELECT      a.student_id, a.student_name, a.student_id_advisor
          FROM        student_advised      parent, student child
          WHERE       parent.student_id_advisor = child.student_id)
          SELECT      student_id, COUNT(student_id)
AS      "students advised"
          FROM        student_advised
          GROUP BY    student_id
          ORDER BY     student_id
```

Getting all course prerequisites or all students being advised queries should be done in an application program where the application CREATEs a temporary table with a simple relational PROJECT:

```
      SELECT      course_id, course_name, course_id_prerequiste, …
```

```
or                student_id, student_name, student_id_advisor, …
      FROM        course
or                student
```

The application program uses CURSOR processing to store the course_id or student_id_advisor as a host variable in the above SQL explosions:

```
      WHERE      c.course_id = :courseid    --: is host variable prefix
```

or:

```
      WHERE      s.student_id_advisor = :studentadvisor
```

The application program transmits the report with the necessary column values and counts on getting a WHENEVER NOT FOUND specifiable by the following SQL within the application program:

```
      EXEC SQL
            WHENEVER NOT FOUND GO TO            symbolic-address
--symbolic address routine transmits report and terminates application
      END-EXEC
```

Processing M:M relationships such as course:book (a course can have zero, one, or many textbooks, and a book can be used in zero, one, or many courses) usually requires an application program. Assume the following child table of COURSE and BOOK:

```
      COURSE_BOOK(COURSE_ID*, BOOK_ID*, title, …)

*      COURSE_ID and BOOK_ID form compound primary key and each is a
foreign key to its parent table (COURSE and BOOK).
```

Assume the following combined report is required:

```
      COURSE NAME                                BOOK TITLE
      PREREQUISITE COURSE NAME                   BOOK TITLE
*******************************
      BOOK TITLE   COURSE NAME                   COURSE NAME
                   COURSE NAME                   COURSE NAME      …
```

The application program's major processing steps are:

1 Determine the highest hierarchical course_id.

2 Execute the embedded single-level explosion illustrated above by substituting a host variable for the c.course_id.

3 Get the textbook(s) required for each course by referencing COURSE_BOOK.

4 Save the required data in temporary course and book tables for transmission on WHENEVER NOT FOUND.

SUMMARY

Common table expressions are useful in simplifying complex SQL by letting DB2 create a temporary result table specifiable in any FROM clause. Recursive common table expressions can be used for many BOM queries without resorting to application programming. Embedding recursive common table expressions lets application programs get the required values without having to do a separate SELECT for each level within a BOM table.

Eric Garrigue Vesely
Principal/Analyst
Workbench Consulting (Malaysia)

DB2 fast close and start – part 2

This month we conclude the code that provides an automated and fast way to stop DB2.

@MDB2001 EDIT MACRO

```
/* REXX */
trace ?o
/*-----   Macro used in REXX @DB2STP0      -----*/
isredit macro
isredit exclude all
isredit find '_|' all
isredit delete x all
isredit end
```

@MDB2036 EDIT MACRO

```
/* REXX */
trace ?o
/*-----   Macro used in REXX @DB2STP2      -----*/
```



```
address ispexec 'VGET (tcanjob)  PROFILE'
isredit macro
isredit exclude '''     '''  51 all
isredit exclude '''TOKEN''' all
isredit exclude ''tcanjob'' all
isredit delete x all
isredit save
isredit end
```

@MDB2037 EDIT MACRO

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

@MDB2039 EDIT MACRO

```
/* REXX */
trace ?o
/*-----   Macro used in : @DB2SDSF   -----*/
isredit macro
isredit exclude all
isredit find EXECUTION  all
isredit delete x all
isredit save
isredit exclude HOLD    all
isredit delete x all
isredit save
isredit end
```

IEBDDIN MEMBER

```
 COPY OUTDD=OUT1,INDD=INP1
```

DB2REXX1 PROC

```
//DB2REXX1 PROC
//COPYTMP  EXEC PGM=IEBCOPY
```

```
//SYSPRINT  DD  SYSOUT=Z
//SYSUT3    DD  UNIT=VIO,SPACE=(CYL,(5,1))
//SYSUT4    DD  UNIT=VIO,SPACE=(CYL,(5,1))
//INP1      DD  DISP=SHR,DSN=ISP.SISPTENU
//OUT1      DD  DSN=&&TENU,DISP=(,PASS),SPACE=(CYL,(1,1,1Ø)),
//          UNIT=WORKA
//SYSIN     DD  DISP=SHR,DSN=user.library(IEBDDIN)
//*  Create a member IEBDDIN into the USER.LIBRARY with the following
//*  string :     COPY OUTDD=OUT1,INDD=INP1
//LAB69     IF (COPYTMP.RC EQ Ø) THEN
//REXXØØ   EXEC PGM=IKJEFTØ1,DYNAMNBR=15Ø
//STEPLIB   DD  DSN=SYS1.DSN51Ø.SDSNLOAD,DISP=SHR
//SYSTSPRT  DD  SYSOUT=X
//SYSTERM   DD  SYSOUT=*
//* ----> CLIST macro and proc user library name  <---- *
//SYSPROC   DD  DSN=user.library,DISP=SHR
//ISPLOG    DD  DUMMY
//ISPPROF   DD  DISP=(,DELETE,DELETE),UNIT=WORKA,SPACE=(CYL,(1,1,1Ø)),
//          DCB=(LRECL=8Ø,BLKSIZE=312Ø,RECFM=FB,DSORG=PO),DSN=&&PROF
//ISPPLIB   DD  DISP=SHR,DSN=ISP.SISPPENU
//ISPMLIB   DD  DISP=SHR,DSN=ISP.SISPMENU
//ISPTLIB   DD  DISP=(OLD,PASS),DSN=&&TENU
//ISPSLIB   DD  DISP=SHR,DSN=ISP.SISPSLIB
//SYSTSIN   DD  DUMMY
//LAB69END  ENDIF
```

SAMPLE BATCH JOB TO SUBMIT DB2 FAST CLOSE PROCEDURE

```
//STOPDSNZ JOB (????ØØØØ),'DB2-Management',CLASS=?,MSGCLASS=X,
//       USER=DBAØØØØ,REGION=3M,MSGLEVEL=(1,1),NOTIFY=&SYSUID
/*JOBPARM BYTES=999999,LINES=9999
//* ------------*       Functions  Procedure         *----------- *
//* ---> STOP/START USER DataBase :                                *
//*           'STOP=no,START=no                                    *
//* ---> STOP DB2 :                                               *
//*           'STOP=YES,START=no'                                  *
//* ---> STOP DB2 + START DB2 :                                   *
//*           'STOP=YES,START=YES'                                 *
//DB2PROC  JCLLIB ORDER=(user.library)
//REXXØØ   EXEC DB2REXX1
//REXXØØ.SYSTSIN  DD  *
  ISPSTART CMD(@DB2STPØ DSNZ,8,STOP=NO,START=NO)
  Sample program DELAYØØ for TSO delay time
  (The sample program has been taken from Issue 115 of MVS Update)
  download from  http://www.xephon.com/cgi-bin/xephon2/tdisplay?/
  m115aØ3.txt
```

Giuseppe Rendano
DB2 Systems Programmer (Italy)

Utility for generating Platinum Fastunload jobs with corresponding IBM load jobs

There are many methods and vendor products to migrate data from one database to another. Typically the data will need to be migrated from a production to a test environment. Some DBAs prefer DSN1COPY, which though very fast has several disadvantages. One disadvantage is trying to maintain the correct and accurate DBID, PSID, and OBID information. There is also the potential to overwrite the wrong dataset if one is not careful. Platinum FastUnload is a convenient product that can generate DSNTIAUL unload datasets from image copies. However, Platinum can generate such jobs only from the most recent full image copies. Also, it is very difficult to set-up the JCL manually for several tables let alone databases. Further, the partitioned datasets require a special set up. This utility, however, simplifies this process and is very flexible and generates the corresponding load jobs for the generated unload jobs. It has been a great time-saver and also made our jobs a lot easier.

This utility is very similar to the RECOVERY utility published in *DB2 Update*, Issues 98 and 99, December 2000 and January 2001. It uses the REXX SQL interface from IBM to access the information from the catalog tables and to build the JCL.

Assumptions and operational considerations:

- Image copies are assumed to be done to a tape dataset.
- The utility will generate the JCL for UNLOADing the data to tape datasets.
- The user must be familiar with writing edit macros and executing them.

Features of the utility:

- It uses a panel for easy and user-friendly input data capture.
- It generates Platinum Fastunload jobs, IBM loadjobs, and loadcontrol dataset lists for running macros to change the load controls.

- The tablespaces to be unloaded are given as a list in a sequential dataset. Each line in this dataset contains the database name (if all tablespaces in the database are to be recovered) or a database and a tablespace name separated by one or more blanks (if a specific tablespace in a database is to be recovered). Blank lines will cause unexpected results.
- All output datasets will be deleted if they exist and new ones will be created. There will be a member for each line in the input dataset.
- The member REPORT in the output dataset for UNLOAD jobs summarizes the input parameters and the status of the processing of each input record.
- It always uses an ICTIME range for identifying the SYSCOPY record to use for recovery. Hence it is imperative that the range is specified so that there is only one full copy record for the object being recovered.
- The unload JCL, load JCL, and load control dataset lists are written into members named as either the database name (if only database name is present) or the tablespace name (if database and tablespace names are provided). All unload jobs are written into one PDS, all load jobs in another PDS, and all load control lists in a third PDS.
- The image copies are assumed to be done to tape devices. Relevant changes may be made to the SQL STMT in PJJFUNLG otherwise.
- Status messages are displayed even as the utility is executing.
- Defaults are generated on the panel wherever possible and the user can override them if required.
- The utility is written for a site that has the production and test subsystems on different machines.
- The panels used by these utilities must be set up in an appropriate dataset and the code in FUNLOD must be modified to ‘libdef’ this dataset to the ISPPLIB.

The REXX SQL interface must be available for use. This was a free download for Version 5.0 for DB2 for OS/390. It is now available along with Version 6.0.

Inputs to the utility are:

1 The source and target DB2 subsystem ID to be used.

2 The dataset containing the list of database names or database and tablespace names to be used for recovery.

3 The ICDATE for recovery.

4 An ICTIME range – a 'From' ICTIME and a 'To' ICTIME value.

5 An output dataset for writing the UNLOAD JCL. This will be allocated by the utility.

6 An output dataset for writing the LOAD JCL. This will be allocated by the utility.

7 An output dataset for writing the loadcontrol dataset lists. This will be allocated by the utility.

8 Two account information fields for use in the JCL job cards.

The output from the utility is three partitioned datasets with the given names and will have one member for each input record member, as explained above.

Important: after successfully executing the Fastunload jobs, and before executing the IBM load jobs, we need to modify the loadcontrols to contain relevant control information like creator id, etc, for the target subsystem. An edit macro can be run on each load control dataset generated by the Platinum Fastunload. Alternatively, the REXX CHGLDCL can be executed on each member of the load control dataset list PDS. This REXX will receive the macro name and the name of a member in the load control list dataset as input and will run the edit macro on each dataset name contained in that member. After successfully processing one line in the input member, a flag 'XX' is appended to it, so that the load control will not be edited by the macro again. The edit macro must be in a dataset concatenated to the SYSPROC dataset, like the CLIST dataset. Since the edit macro

is site-specific, it is not provided here.

This utility (FUNLOD) generates the members in the load control list dataset along with the unload and load members and in the same fashion.

This utility has two REXX programs and two panels and an additional REXX program CHGLDCL.

- FUNLOD – the main program that performs panel display, validations, and calls the program PJJFUNLG.
- PJJFUNLG – the program, called by the main program, that performs all the processing.
- FUNLPAN – the main data capture panel.
- FUNLPAN2 – the panel for displaying error messages.
- CHGLDCL – a separate REXX to run an edit macro (tailored specifically to your installation) on the loadcontrols to change their information as appropriate.

Site-specific information is marked in the code as SITESPEC. At our installation the production systems are on one machine and the test systems on another machine. You may need to modify the code according to your site.

FUNLOD

```
/* rexx */
/*********************************************************************/
/* Panel driven tool for generating JCL for unload jobs from         */
/* image copies using Platinum's Fastunload and corresponding load   */
/* jobs using IBM loads. The load controls must be altered suitably  */
/* Invocation: TSO FUNLOD                                            */
/*********************************************************************/
trace o
clear
address tso "ispexec  vget (zsysid)"
p_sysid=zsysid
if p_sysid = 'PROD' then
   MC = 'PROD'
else
   MC = 'TEST'
pref =strip(sysvar(syspref))
```

```
cd = date(U)
us_date = substr(cd,7,2)||substr(cd,1,2)||substr(cd,4,2)
ZWINTTL = 'UNLOAD & LOAD FROM IMAGE COPIES'
DBLST = pref||'.'userid()||'.DBNLST'
ACCTNUM='123AB456'
ACCTNUL='123CD456'
cur_date = date(S)-1
cur_date = substr(cur_date,3)
ICDATE = cur_date
ICT1 = '190000'
ICT2 = '235900'
EDFLG= 'Y'
RECTYP = 'F'
OUTUNL = pref||'.'||userid()||'.UNLOUT.D'||us_date
OUTLOD = pref||'.'||userid()||'.LODOUT.D'||us_date
LDCLST = pref||'.'||userid()||'.LDCLST.D'||us_date
BEGIN:
/*  Specify an appropriate panel dataset where the panels FUNLPAN */
/*  and FUNLPAN2 will be copied                                   */
"ISPEXEC LIBDEF ISPPLIB DATASET ID('SYS1.PROD.PANELS')"
"ISPEXEC ADDPOP ROW(2) COLUMN(10)"
"ISPEXEC DISPLAY PANEL(FUNLPAN)"
"ISPEXEC REMPOP ALL "
clear
Upper DBLST
upper EDFLG
upper ICDATE
upper ICT1; upper ICT2; upper JCLMDL; upper OUTDS; upper JCLMEM
upper SSID
if PF3 = 'EXIT' then
  exit(0)
CHK_DSN:
DBLST = strip(DBLST,Leading,"'")
z = outtrap("zlst.","*")
x = SYSDSN("'"DBLST"'")
z = outtrap("OFF")
if x = 'OK' then
   nop
else
do
   ERRMSG = 'Dataset not found ...'
   ERRCOL = DBLST
   "ISPEXEC ADDPOP ROW(4) COLUMN(6)"
   "ISPEXEC DISPLAY PANEL(FUNLPAN2)"
   "ISPEXEC REMPOP ALL "
   signal BEGIN
end
CHK_ICDATE:
D_yy = substr(ICDATE,1,2)
D_mm = substr(ICDATE,3,2)
```

```
D_dd = substr(ICDATE,5,2)
if (D_mm > 13) | (D_dd > 31) then
do
   ERRMSG = 'Invalid date specified '
   ERRCOL = ICDATE
   "ISPEXEC ADDPOP ROW(4) COLUMN(6)"
   "ISPEXEC DISPLAY PANEL(FUNLPAN2)"
   "ISPEXEC REMPOP ALL "
    signal BEGIN
end
cur_date = date(S)
cur_date = substr(cur_date,3)
if ICDATE > cur_date then
do
   ERRMSG = 'Date must be < =  'cur_date
   ERRCOL = ICDATE
   "ISPEXEC ADDPOP ROW(4) COLUMN(12)"
   "ISPEXEC DISPLAY PANEL(RCVR51)"
   "ISPEXEC REMPOP ALL "
    signal BEGIN
end
CHK_ICT1:
T_hh = substr(ICT1,1,2)
T_mm = substr(ICT1,3,2)
T_ss = substr(ICT1,5,2)
if (T_hh > 23) | (T_mm > 59) | (T_ss > 59) then
do
   ERRMSG = 'Invalid time specified '
   ERRCOL = ICT1
   "ISPEXEC ADDPOP ROW(4) COLUMN(6)"
   "ISPEXEC DISPLAY PANEL(FUNLPAN2)"
   "ISPEXEC REMPOP ALL "
    signal BEGIN
end
CHK_ICT2:
T_hh2 = substr(ICT2,1,2)
T_mm2 = substr(ICT2,3,2)
T_ss2 = substr(ICT2,5,2)
if ((T_hh2 > 23) | (T_mm2 > 59) | (T_ss2 > 59))  then
do
   ERRMSG = 'Invalid time specified '
   ERRCOL = ICT2
   "ISPEXEC ADDPOP ROW(4) COLUMN(6)"
   "ISPEXEC DISPLAY PANEL(FUNLPAN2)"
   "ISPEXEC REMPOP ALL "
    signal BEGIN
end
CHK_ICTS:
if (T_hh2 < T_hh) | (T_hh2 = T_hh & T_mm2 < T_mm) then
do
```

```
   ERRMSG = 'ICTIMES incorrect  '
   ERRCOL = ICT1||' < 'ICT2
   "ISPEXEC ADDPOP ROW(4) COLUMN(6)"
   "ISPEXEC DISPLAY PANEL(FUNLPAN2)"
   "ISPEXEC REMPOP ALL "
    signal BEGIN
end
CHKSSID:
address tso "ispexec  vget (zsysid)"
p_sysid=zsysid
sid = SSID
ssid_err = Ø
 if p_sysid  = 'PROD' then
    select
      when sid = 'PROD'
         then do
            nop
         end
      otherwise
         ssid_err =1
 end
 if p_sysid  = '819Ø' then
   select
      when sid = 'DBT1'
         then do
            nop
         end
      when sid = 'DBT2'
         then do
            nop
         end
      when sid = 'DBT3'
         then do
            nop
         end
      otherwise
         ssid_err =1
   end
if ssid_err = 1 then
do
   ERRMSG = 'Invalid SSID for '||MC
   ERRCOL = SSID
   "ISPEXEC ADDPOP ROW(4) COLUMN(6)"
   "ISPEXEC DISPLAY PANEL(FUNLPAN2)"
   "ISPEXEC REMPOP ALL "
    signal BEGIN
end
CHKEDT:
if EDFLG = 'Y' then
do
```

```
   ADDRESS ISPEXEC "EDIT  DATASET('"DBLST"')"
end
if ACCTNUL = ' ' then
  ACCTNUL = ACCTNUM
cmd = 'address tso "PJJFUNLG" SSID DBLST ICDATE ICT1 ICT2 '
cmd = cmd||' ACCTNUM OUTUNL OUTLOD LDCLST TSID '
interpret cmd
say 'Press Enter to quit...'; pull temp
exit(0)
```

PJJFUNLG

```
/* rexx */
/*********************************************************************/
/* PJJFUNLG - called by FUNLOD after receiving all input            */
/*  through the panel displayed by FUNLOD                            */
/* Site-specific information is marked by ***SITESPEC***            */
/*  Uses REXX SQL interface provided by IBM                          */
/*********************************************************************/
trace o
clear
PARSE UPPER ARG P_ssid P_dblst P_icdate P_ict1 P_ict2 P_act,
     P_out P_lout P_ldcn P_tssid
pref =strip(sysvar(syspref))
sid = strip(P_ssid)
tssid = strip(P_tssid)
/* say  P_ssid P_dblst P_icdate P_ict1 P_ict2 P_jcl P_mem P_out */
I_lstdsn = strip(P_dblst)
I_icdate = strip(P_icdate)
I_btime = strip(P_ict1)
I_etime = strip(P_ict2)
I_jclds = strip(P_jcl)
I_jclmem = strip(P_mem)
ods_name = strip(P_out)
lds_name = strip(P_lout)
ldcn_name = strip(P_ldcn)
P_act    = strip(P_act)
q = 0  /* counter for ldcntl list  */
cd = date(U)
us_date = substr(cd,7,2)||substr(cd,1,2)||substr(cd,4,2)
Call GETDBLST
Call P000_MAIN
exit(0)
GETDBLST:
"ALLOCATE DD(lstdd) DSN('"I_lstdsn"') REUSE SHR"
if  rc>0 then
do
   say  'Failed during allocation of 'I_lstdsn
   pull temp
```

```
   exit(8)
end
"execio * DISKR lstdd (FINIS STEM dbl."
return
P000_MAIN:
pref = sysvar(syspref)
pref = strip(pref)
P_ssid = strip(sid)
xx = outtrap("zap.","*")
address tso "delete '"ods_name"'"
address tso "delete '"lds_name"'"
address tso "delete '"ldcn_name"'"
xx = outtrap("OFF")
/*   allocating   unload jobs dataset      */
address tso "alloc f(rpds) new unit(hsm) space(5,10)",
          "cyl dir(50) reuse dsname('"ods_name"')",
          "dsorg(po) blksize(3120) lrecl(80) recfm(f b)"
if  rc>0 then
do
   say  'Failed during allocation of 'ods_name
   pull temp
   exit(8)
end
/*   allocating  load jobs dataset  */
address tso "alloc f(lpds) new unit(hsm) space(5,10)",
          "cyl dir(50) reuse dsname('"lds_name"')",
          "dsorg(po) blksize(3120) lrecl(80) recfm(f b)"
if  rc>0 then
do
   say  'Failed during allocation of 'lds_name
   pull temp
   exit(8)
end
/*   allocating  loadcntl list dataset  */
address tso "alloc f(ldcn) new unit(hsm) space(2,1)",
          "cyl dir(50) reuse dsname('"ldcn_name"')",
          "dsorg(po) blksize(3120) lrecl(80) recfm(f b)"
if  rc>0 then
do
   say  'Failed during allocation of 'ldcn_name
   pull temp
   exit(8)
end
s = 0;
Call PERFCONN
do i = 1 to dbl.0
   Parse Var dbl.i dbname 9 tsname 18 rest
   I_dbname=strip(dbname);
   I_tsname=strip(tsname);
   Call Step1
```

```
end
Call Step3
Call PERFDISC
say 'Doing cleanup...'
xx = outtrap("zap.","*")
address tso "free f(lstdd)"
address tso "FREE ddname(rpds)"
address tso "FREE ddname(lpds)"
xx = outtrap("OFF")
say; say 'Completed processing..'
say 'Refer member REPORT in 'ods_name
say 'The load jobs are in 'lds_name
say 'The load cntl datasets are in 'ldcn_name
return
Step1:
STMT = "   SELECT  A.DBNAME,A.TSNAME, "  ,
      "      DIGITS(A.DSNUM) AS PART, "  ,
      "      DIGITS(A.FILESEQNO) AS SEQNO,  " ,
      "      A.DSNAME,  B.NAME, B.CREATOR  " ,
      "   FROM   SYSIBM.SYSCOPY A, SYSIBM.SYSTABLES B  " ,
      "   WHERE A.DBNAME =    '"I_dbname"'"
if I_tsname ¬= '' then
do
   STMT = STMT || " AND  A.TSNAME = '"I_tsname"'"
end
else
     nop
STMT = STMT||"   AND A.ICDATE = '"I_icdate"'" ,
      "   AND A.ICTIME BETWEEN '"I_btime"' AND '"I_etime"'" ,
      "   AND A.ICTYPE = 'F' " ,
      "   AND A.ICUNIT = 'T'  " ,
      "   AND A.ICBACKUP = ' ' "  ,
      "   AND A.DBNAME = B.DBNAME   " ,
      "   AND A.TSNAME = B.TSNAME  "  ,
      "   ORDER BY 1,2,5,3,4    "
Call PROCQRY
Call Step2
return
Step2:
k = Ø; z = Ø ; p = Ø ; q=Ø
eof=Ø; osn = Ø
len= length(I_dbname);
If strip(I_tsname) = '' then
   memname = I_dbname
else
   memname = I_tsname
I_dbname = strip(I_dbname)
I_tsname = strip(I_tsname)
Do while length(I_dbname) < 8
   I_dbname = ' '||I_dbname
```



```
end
Do while length(I_tsname) < 8
   I_tsname = ' '||I_tsname
end
say 'Processing subset 'Memname '...'
prevts = ' '
prevtb = ' '
co = ','
ic=Ø
first = 1
Do until (SQLCODE ¬= Ø )
   db = strip(A1)
   ts = strip(A2)
   prt = strip(A3)
   fseq = strip(A4)
   dsn = strip(A5)
   tbn = strip(A6)
   tbcr= strip(A7)
   if  first = 1  then
   do
      prevts = (ts)
      prevtb = (tbn)
      Call  BILDSTEP
      first = Ø
   end
   if  prevts = ts then
   do
      if (prt > 1)  then
      do
         k=k+1
         out.k = "//         DD  DSN="||dsn||","
         k=k+1
         out.k = "//             DISP=(OLD,PASS),VOL=(,RETAIN) "
      end
      else
         nop
   end
   else   /*  prevts not equal to ts  */
   do
      k=k+1
      out.k = "//SYSIN    DD * "
      Call  ADDCTRL
      Call  BILDLOAD
      prevts = strip(ts)
      prevtb = strip(tbn)
      Call  BILDSTEP
   end
ADDRESS DSNREXX "EXECSQL FETCH C1 INTO :A1, :A2, :A3, :A4, :A5,",
                ":A6, :A7"
/*******************************************************/
```

```
/*
   IF  SQLCODE = Ø  Then  Do
          Line = ' '
          Line = Line||A1||A2||A3||A4||A5||A6||A7
          say Line
   end
*/
/*********************************************************/
end
ADDRESS DSNREXX "EXECSQL CLOSE C1"
/*  complete the rest of the build for last row */
   k=k+1
   out.k = "//SYSIN    DD * "
   Call  ADDCTRL
   Call  BILDLOAD
Call  FILLJCL
Call  FILLJCLD
Call  WRITEMEM
return
BILDSTEP:
osn = osn+1;  udsn = ''; cdsn = ''
loc = pos('L',dsn,1)
loc1 = loc-1;loc2= loc+1
udsn = substr(dsn,1,loc1)||'U'||substr(dsn,loc2)
cdsn = substr(dsn,1,loc1)||'C'||substr(dsn,loc2)
q = q+1
ldlst.q = cdsn
do while length(osn) < 4
   osn = 'Ø'||osn
end
k=k+1
out.k ="//UTIL"||osn||" EXEC PGM=PTLDRIVM,"
k=k+1
out.k ="//           PARM='EP=UTLGLCTL/"||sid||"',  "
k=k+1
out.k ="//           REGION=ØM                                  "
k=k+1
out.k ="//STEPLIB  DD DSN=ABCD.PLATINUM.LOADLIB,DISP=SHR          "
k=k+1
out.k ="//         DD DSN=ABCD."||sid||".DSNLOAD,DISP=SHR  "
k=k+1
out.k ="//PTILIB   DD DSN=ABCD.PLATINUM.LOADLIB,DISP=SHR          "
k=k+1
out.k ="//         DD DSN=ABCD."||sid||".DSNLOAD,DISP=SHR  "
k=k+1
out.k ="//PTIPARM  DD DSN=ABCD.PLATINUM.PARMLIB,DISP=SHR          "
k=k+1
out.k ="//PTIXMSG  DD DSN=ABCD.PLATINUM.XMESSAGE,DISP=SHR         "
k=k+1
out.k ="//PTIMSG   DD SYSOUT=*                                    "
```

```
k=k+1
out.k ="//*                                                              "
k=k+1
out.k ="//SYSOUT   DD SYSOUT=*                                          "
k=k+1
out.k ="//SYSUDUMP DD SYSOUT=*                                          "
k=k+1
out.k ="//ABNLIGNR DD DUMMY                 SUPPRESS ABENDAID DUMPS"
k=k+1
out.k ="//*                                                              "
k=k+1
out.k ="//ST01WK01 DD UNIT=SYSDA,SPACE=(CYL,(300,200))                  "
k=k+1
out.k ="//ST01WK02 DD UNIT=SYSDA,SPACE=(CYL,(300,200))                  "
k=k+1
out.k ="//ST01WK03 DD UNIT=SYSDA,SPACE=(CYL,(300,200))                  "
k=k+1
out.k ="//ST01WK04 DD UNIT=SYSDA,SPACE=(CYL,(300,200))                  "
k=k+1
out.k ="//ST02WK01 DD UNIT=SYSDA,SPACE=(CYL,(300,200))                  "
k=k+1
out.k ="//ST02WK02 DD UNIT=SYSDA,SPACE=(CYL,(300,200))                  "
k=k+1
out.k ="//ST02WK03 DD UNIT=SYSDA,SPACE=(CYL,(300,200))                  "
k=k+1
out.k ="//ST02WK04 DD UNIT=SYSDA,SPACE=(CYL,(300,200))                  "
k=k+1
out.k ="//ST03WK01 DD UNIT=SYSDA,SPACE=(CYL,(300,200))                  "
k=k+1
out.k ="//ST03WK02 DD UNIT=SYSDA,SPACE=(CYL,(300,200))                  "
k=k+1
out.k ="//ST03WK03 DD UNIT=SYSDA,SPACE=(CYL,(300,200))                  "
k=k+1
out.k ="//ST03WK04 DD UNIT=SYSDA,SPACE=(CYL,(300,200))                  "
k=k+1
out.k ="//ST01MSG  DD SYSOUT=*                                          "
k=k+1
out.k ="//ST02MSG  DD SYSOUT=*                                          "
k=k+1
out.k ="//ST03MSG  DD SYSOUT=*                                          "
k=k+1
out.k ="//*                                                              "
k=k+1
out.k ="//SYSCTL01 DD DSN="||cdsn||","
k=k+1
out.k ="//            DISP=(,CATLG,DELETE),SPACE=(CYL,(1,1)),UNIT=HSM "
k=k+1
out.k ="//SYSREC01 DD DISP=(,CATLG,DELETE),LABEL=("osn",SL,RETPD=30),"
k=k+1
out.k ="//            UNIT=3490,         "
```

```
k=k+1
out.k= "//            DSN="||udsn||","
k=k+1
posn=osn-1
do while length(posn) < 4
   posn = '0'||posn
end
if osn = 1 then
   out.k="//            VOL=(,RETAIN)                          "
else
   out.k="//            VOL=(,RETAIN,REF=*.UTIL"||posn||".SYSREC01)  "
k=k+1
out.k ="//SYSIMAG  DD DSN="||dsn||","
k=k+1
out.k ="//            DISP=(OLD,PASS),VOL=(,RETAIN)         "
/*  code to build the stem for delete steps  */
p = p+1
del.p = "   DELETE "||udsn||" PURGE SCRATCH "
p = p+1
del.p = "   DELETE "||cdsn
return
/**    end  of  BILDSTEP    **/
ADDCTRL:
k=k+1
out.k = "FASTUNLOAD"
k=k+1
out.k = "DISCARDS 1                "
k=k+1
out.k = "EXCP NO                   "
k=k+1
out.k = "INPUT-FORMAT IMAGECOPY    "
k=k+1
out.k = "IO-BUFFERS 60             "
k=k+1
out.k = "VSAM-BUFFERS 96           "
k=k+1
out.k = "LOAD-CONTROL DB2LOAD      "
k=k+1
out.k = "OUTPUT-FORMAT DSNTIAUL    "
k=k+1
out.k = "SHRLEVEL REFERENCE        "
k=k+1
out.k = "SORTSIZE MAX              "
k=k+1
out.k = "SORTNUM  4                "
k=k+1
out.k = "ESTIMATED-ROWS 20000000   "
k=k+1
out.k = "SQL-ACCESS  NONE          "
k=k+1
```

```
/*  tbname = substr(prevts,1,6)||"TB"  */
out.k = "SELECT * FROM "||tbcr||"."||prevtb||";      "
k=k+1
out.k = "/*                          "
k=k+1
out.k = "//*                         "
return
BILDLOAD:
tsnm = substr(prevts,3,4)
z = z+1
lod.z = "//LOAD"||osn||" EXEC PGM=DSNUTILB,"
lprm  = "PARM=("||tssid||",'LD"||substr(prevts)||"'),"
lod.z = lod.z||lprm
z = z+1
lod.z = "//*LOAD"||osn||" EXEC PGM=DSNUTILB,"
lod.z = lod.z||lprm||"RESTART(PHASE)),"
z = z+1
lod.z = "//            COND=(4,LT),REGION=0M              "
z = z+1
lod.z = "//STEPLIB  DD DISP=SHR,DSN=SGDP."||tssid||".DSNEXIT  "
z = z+1
lod.z = "//         DD DISP=SHR,DSN=SGDP."||tssid||".DSNLOAD  "
z = z+1
lod.z = "//DSNTRACE DD SYSOUT=*                                 "
z = z+1
lod.z = "//SYSABEND DD SYSOUT=D,HOLD=YES                        "
z = z+1
lod.z = "//SYSPRINT DD SYSOUT=*                                 "
z = z+1
lod.z = "//SYSUDUMP DD SYSOUT=*                                 "
z = z+1
lod.z = "//UTPRINT  DD SYSOUT=*                                 "
z = z+1
lod.z = "//SYSOUT   DD SYSOUT=*                                 "
z = z+1
lod.z = "//SORTWK01 DD
DISP=(NEW,DELETE,DELETE),SPACE=(CYL,(500,300),RLSE),"
z = z+1
lod.z = "//            UNIT=SYSDA                               "
z = z+1
lod.z = "//SORTWK02 DD
DISP=(NEW,DELETE,DELETE),SPACE=(CYL,(500,300),RLSE),"
z = z+1
lod.z = "//            UNIT=SYSDA                             "
z = z+1
lod.z = "//SORTWK03 DD
DISP=(NEW,DELETE,DELETE),SPACE=(CYL,(500,300),RLSE),"
z = z+1
lod.z = "//            UNIT=SYSDA                             "
z = z+1
```

```
lod.z = "//SORTWKØ4 DD
DISP=(NEW,DELETE,DELETE),SPACE=(CYL,(5ØØ,3ØØ),RLSE),"
z = z+1
lod.z = "//            UNIT=SYSDA                                    "
z = z+1
lod.z = "//SYSERR   DD DSN="||pref||"."||userid()||"."||tsnm||".SYSERR,"
lod.z = lod.z||"UNIT=SYSDA,"
z = z+1
lod.z = "//
DISP=(NEW,DELETE,CATLG),SPACE=(CYL,(5ØØ,3ØØ),RLSE) "
z = z+1
lod.z = "//SYSMAP   DD DSN="||pref||"."||userid()||"."||tsnm||".SYSMAP,"
lod.z = lod.z||"UNIT=SYSDA,"
z = z+1
lod.z = "//
DISP=(NEW,DELETE,CATLG),SPACE=(CYL,(5ØØ,3ØØ),RLSE)"
z = z+1
lod.z = "//SYSUT1   DD DSN="||pref||"."||userid()||"."||tsnm||".SYSUT1,"
lod.z = lod.z||"UNIT=SYSDA,"
z = z+1
lod.z = "//
DISP=(NEW,DELETE,CATLG),SPACE=(CYL,(5ØØ,3ØØ),RLSE)"
z = z+1
lod.z = "//SORTOUT  DD
DSN="||pref||"."||userid()||"."||tsnm||".SORTOUT,"
lod.z = lod.z||"UNIT=SYSDA,"
z = z+1
lod.z = "//
DISP=(NEW,DELETE,CATLG),SPACE=(CYL,(5ØØ,3ØØ),RLSE)"
z = z+1
lod.z = "//*************************************              "
z = z+1
lod.z = "//*********LOAD  FILES      **************              "
z = z+1
lod.z = "//*************************************              "
z = z+1
lod.z = "//SYSRECØ1 DD DISP=SHR,                                "
z = z+1
lod.z = "//            DSN="||udsn||",                          "
z = z+1
lod.z="//            VOL=(,RETAIN)                              "
z = z+1
lod.z = "//SYSIN    DD DISP=SHR,                                "
z = z+1
lod.z = "//            DSN="||cdsn||"                           "
z = z+1
lod.z = "//********  "
return
FILLJCL:
jcl.1 = "//"||substr(memname,1,6)||"UL JOB ("||P_act||"),"
```

```
jcl.1 = jcl.1||"'"||memname||"FASTUL',TIME=144Ø,"
/*    ****SITESPEC****                                              */
jcl.2 = "//           CLASS=B,MSGCLASS=X,REGION=2ØM,NOTIFY=&SYSUID, "
jcl.3 = "//           LINES=9999                                     "
jcl.4 = "//*     "
/*      ****SITESPEC****                                            */
jcl.5 = "/*JOBPARM SYSAFF=PROD                                     "
jcl.6 = "//*                                                       "
/*  code to add delete step   */
jcl.7 = "//*****************************************************"
jcl.8 = "//** DELETE NEW DATASETS                                 **"
jcl.9 = "//** ALL DATASETS WILL HAVE THE SAME TIMESTAMP           **"
jcl.1Ø= "//**    AS THE IMAGE COPY                                 **"
jcl.11= "//*****************************************************"
jcl.12= "//S1DEL1  EXEC PGM=IDCAMS,COND=(4,LT)                      "
jcl.13= "//SYSPRINT DD SYSOUT=*                                     "
jcl.14= "//SYSIN    DD *                                            "
jjl.1 = "//* ---------------------------------------------    "
jjl.2 = "//*       FAST UNLOAD JOB                            "
jjl.3 = "//*                                                  "
jjl.4 = "//*       GENERATED USING FUNLOD  on "||us_date||"    "
jjl.5 = "//* USING "||I_lstdsn||" "
jjl.6 = "//* ---------------------------------------------    "
return
FILLJCLD:
lcl.1 = "//"||substr(memname,1,6)||"LD JOB (427AR968),"
lcl.1 = lcl.1||"'"||memname||"IBMLOD',TIME=144Ø,"
lcl.2 = "//           CLASS=B,MSGCLASS=T,REGION=2ØM,NOTIFY=&SYSUID, "
lcl.3 = "//           LINES=9999                                 "
lcl.4 = "//*                                                     "
lcl.5 = "/*JOBPARM SYSAFF=TEST                                   "
lcl.6 = "//*                                                     "
lcl.7 = "//* ---------------------------------------------    "
lcl.8 = "//*       IBM LOAD JOB                               "
lcl.9 = "//*                                                  "
lcl.1Ø= "//*       GENERATED USING JJFUNL  on "||us_date||"        "
lcl.11= "//* using "||I_lstdsn||" "
lcl.12= "//* ---------------------------------------------    "
return
************************************************************
WRITEMEM:
   ods_mem = ods_name||"("||memname||")"
   lds_mem = lds_name||"("||memname||")"
   ldcn_mem = ldcn_name||"("||memname||")"
   address tso "alloc f(outdd) mod dsname('"ods_mem"')"
   address tso "alloc f(loutdd) mod dsname('"lds_mem"')"
   address tso "alloc f(ldcndd) mod dsname('"ldcn_mem"')"
   address tso "execio *  DISKW outdd (stem jcl.  "
   p = p+1
   del.p = "  IF MAXCC LE 8 THEN SET MAXCC EQ Ø "
```

```
   p = p+1
   del.p = "/*                                    "
   address tso "execio *  DISKW outdd (stem del.  "
   address tso "execio *  DISKW outdd (stem jjl.  "
   address tso "execio *  DISKW outdd (stem out. FINIS "
   CC = RC
   address tso "execio *  DISKW loutdd (stem lcl.  "
   address tso "execio *  DISKW loutdd (stem lod. FINIS "
   LC = RC
   address tso "execio *  DISKW ldcndd (stem ldlst. FINIS "
   LDC = RC
   drop out.
   drop lod.
   drop jcl.
   drop jjl.
   drop lcl.
   drop del.
   drop ldlst.
   xx = outtrap("zap.","*")
   address tso "FREE ddname(outdd)"
   address tso "FREE ddname(loutdd)"
   address tso "FREE ddname(ldcndd)"
   xx = outtrap("OFF")
   s = s+1
   sno=s
   do while length(sno) < 3
      sno = ' '||sno
   end
   if CC = 0 then
   do
      rpt.s = sno||' '||I_dbname||' '||I_tsname||' '||memname
      rpt.s = rpt.s||'  successfully written'
   end
   else
   do
      rpt.s = sno||' '||I_dbname||' '||I_tsname||' '||Memname
      rpt.s = rpt.s||'  failed writing '
   end
return
Step3:
rpt_mem = ods_name||"(REPORT)"
address tso "alloc f(rptdd) mod dsname('"rpt_mem"')"
hdr.1 = 'ICDATE used ...'I_icdate
hdr.2 = 'ICTIMES between 'I_btime' and 'I_etime
hdr.3 = 'Input dataset 'I_lstdsn
hdr.4 = 'NO  DBNAME   TSNAME   MEMBER    STATUS'
hdr.5 = '-------------------------------------------------------------'
address tso "execio *  DISKW rptdd (stem hdr.  "
address tso "execio *  DISKW rptdd (stem rpt. FINIS "
address tso "FREE ddname(rptdd)"
```

```
xx = outtrap("zap.","*")
xx = outtrap("OFF")
return
PERFCONN:
/* Set up STEPLIBs                                    */
/*                                                    */
ADDRESS TSO
lib.Ø = 2
lib.1 = "ABCD."||sid||".DSNLOAD"
lib.2 = "ABCD."||sid||".DSNEXIT"
L_Dsname = ""
  do i = 1 to lib.Ø
    L_Dsname = L_Dsname||"'"||lib.i||"' "
  end
/*  address tso "ALLOC F(STEPLIB) DA("L_Dsname") SHR REU "   */
  "STEPLIB DA("L_Dsname") SHR "
'SUBCOM  DSNREXX'            /* Is host command env avlbl ?  */
IF  RC then                  /*  If not, then add it         */
    S_RC = RXSUBCOM('ADD','DSNREXX','DSNREXX')
ADDRESS  DSNREXX             /* exec all further cmds in DSNREXX */
ADDRESS DSNREXX "CONNECT" sid
Say 'Connect RC ...' RC
return
PROCQRY:
ADDRESS DSNREXX "EXECSQL DECLARE C1 CURSOR FOR S1"
/* Say 'Declare SQLCODE = ' SQLCODE */
if SQLCODE ¬= Ø then
do
   Call SQLCA
   exit
end
ADDRESS DSNREXX "EXECSQL PREPARE S1 FROM :STMT"
/*  Say 'Prepare SQLCODE = ' SQLCODE    */
if SQLCODE ¬= Ø then
do
   if  SQLCODE < Ø then
   do
      Call SQLCA
      exit
   end
end
/*  Say 'opening C1 '                        */
ADDRESS DSNREXX "EXECSQL OPEN C1"
/*  Say 'Open SQLCODE = ' SQLCODE            */
if SQLCODE ¬= Ø then
   Call SQLCA
ADDRESS DSNREXX "EXECSQL FETCH C1 INTO :A1, :A2, :A3, :A4, :A5,",
                ":A6, :A7"
/*
IF  SQLCODE = Ø  Then  Do
```

```
       Line = ' '
       Line = Line||A1||A2||A3||A4||A5||A6||A7
       say Line
end
*/
if SQLCODE ¬= Ø then
do
   Call ERRPT
   say 'No rows retrieved, re-enter data...'
   exit(Ø)
end
/*********************************************************/
/*
IF  SQLCODE = Ø  Then  Do
       Line = ' '
       Line = Line||A1||A2||A3||A4||A5||A6||A7
       say Line
end
*/
/*********************************************************/
return
PERFDISC:
ADDRESS  DSNREXX " DISCONNECT"
Say 'Disconnect RC ...' RC
if SQLCODE ¬= Ø then
do
   Call SQLCA
   exit
end
S_RC = RXSUBCOM('DELETE','DSNREXX','DSNREXX')
Say 'Delete RC ... ' S_RC
return
ERRPT:
do
   do while length(I_dbname) < 8
      I_dbname = ' '||I_dbname
   end
   do while length(I_tsname) < 8
      I_tsname = ' '||I_tsname
   end
   s = s+1
   sno=s
   do while length(sno) < 3
      sno = ' '||sno
   end
   rpt.s = sno||' '||I_dbname||' '||I_tsname||' '||'--------  '
   rpt.s = rpt.s||' - Query did not retrieve any rows'
end
return
SQLCA:
```

```
TRACE O
SAY 'SQLCODE ='SQLCODE
SAY 'SQLERRMC ='SQLERRMC
SAY 'SQLERRP ='SQLERRP
SAY 'SQLERRD ='SQLERRD.1','
|| SQLERRD.2','
|| SQLERRD.3','
|| SQLERRD.4','
|| SQLERRD.5','
|| SQLERRD.6
SAY 'SQLWARN ='SQLWARN.Ø','
|| SQLWARN.1','
|| SQLWARN.2','
|| SQLWARN.3','
|| SQLWARN.4','
|| SQLWARN.5','
|| SQLWARN.6','
|| SQLWARN.7','
|| SQLWARN.8','
|| SQLWARN.9','
|| SQLWARN.1Ø
SAY 'SQLSTATE='SQLSTATE
SAY 'SQLCODE ='SQLCODE
say  'SQLERRMC ='SQLERRMC';' ,
|| 'SQLERRP ='SQLERRP';' ,
|| 'SQLERRD ='SQLERRD.1',',
            || SQLERRD.2',',
            || SQLERRD.3',',
            || SQLERRD.4',',
            || SQLERRD.5',',
            || SQLERRD.6';' ,
|| 'SQLWARN ='SQLWARN.Ø',',
            || SQLWARN.1',',
            || SQLWARN.2',',
            || SQLWARN.3',',
            || SQLWARN.4',',
            || SQLWARN.5',',
            || SQLWARN.6',',
            || SQLWARN.7',',
            || SQLWARN.8',',
            || SQLWARN.9',',
            || SQLWARN.1Ø';' ,
            || 'SQLSTATE='SQLSTATE';'
return
```

FUNLPAN

```
)ATTR
+ TYPE(TEXT) COLOR(WHITE)
```

```
{ TYPE(OUTPUT) COLOR(RED)
¬ TYPE(TEXT) COLOR(GREEN) INTENS(LOW) HILITE(REVERSE)
_ TYPE(INPUT) COLOR(WHITE) INTENS(LOW) HILITE(REVERSE)
| TYPE(OUTPUT) INTENS(HIGH)  CAPS(OFF)  PAD(' ')
)BODY
+ System: {MC  +
+ Source SSID :_SSID+     Target SSID :_TSID+
+
+ DBNAME/TSNAME List PDS :_DBLST                                        +
+ Edit this dataset (Y/N):_Z+
+
+ ICDATE value to unload from          :_ICDATE+(yymmdd)
+ ICTIME value AFTER  which to unload  :_ICT1  +(hhmmss)
+ ICTIME value BEFORE which to unload  :_ICT2  +(hhmmss)
+
+ Account no. for Unload jobs :_ACCTNUM +
+ Account no. for Load jobs   :_ACCTNUL + (Blank if same)
+
+ Dataset for Unload JCLs:_OUTUNL                                        +
+ Dataset for Load JCLs  :_OUTLOD                                        +
+ Dataset for LDCNTL list:_LDCLST                                        +
+
   |ERRMSG                    |ERRCOL
+
¬F3 - End +
+
)INIT
  .ZVARS= 'EDFLG'
)PROC
   if (.PFKEY = 'PFØ3') &PF3 = EXIT
   VER(&SSID,NB,LIST,DBT3,DBP2,DBT1,DBT2,DBP4,DBP5,DBP6,DBP1,DBP3)
   VER(&TSID,NB,LIST,DBT3,DBP2,DBT1,DBT2,DBP4,DBP5,DBP6,DBP1,DBP3)
   VER(&DBLST,NB)
   VER(&EDFLG,NB,LIST,Y,N)
   VER(&ICDATE,NB)
   VER(&ICT1,NB)
   VER(&ICT2,NB)
   VER(&ACCTNUM,NB)
   VER(&OUTUNL,NB)
   VER(&OUTLOD,NB)
   VER(&LDCLST,NB)
)END
```

FUNLPAN2

```
)ATTR
+ TYPE(TEXT) COLOR(WHITE)
)BODY WINDOW (6Ø,5)
+
```

```
+ &ERRMSG                    :+ &ERRCOL
+
+  Enter to continue ...
+
)INIT
)PROC
)END
```

CHGLDCL

```
/*   rexx  */
/*  CHGLDCL  - REXX to run an edit macro on a list of datasets       */
/*  The edit macro must be in a dataset concatenated to your         */
/*  SYSPROC dataset.  The edit macro must be suitably altered        */
/*  to meet your requirements                                        */
/*  The first input will be a dataset containing a list of datasets  */
/*  on which to run the edit macro. This will normally be a member   */
/*  in the dataset containing the list of load-control datasets      */
trace o
clear
say 'Give the dataset set list to run the macro on..'
parse pull I_runlst
if I_runlst = '' then
do
   say 'Input dataset not provided. Ending rexx'
   exit(8)
end
say 'Give the name of the macro member to run..'
parse pull I_macmem
I_runlst = strip(I_runlst,B," ")
I_runlst = strip(I_runlst,B,"'")
I_macmem = strip(I_macmem)
I_macmem = strip(I_macmem,B," ")
"ALLOCATE DD(INDD) DSN('"I_runlst"') REUSE SHR"
do forever
   address tso  "execio 1 DISKRU INDD "
   if rc=2 then leave
   pull inrec1
   instrng =strip(inrec1)
   parse var instrng dsname  flg  dummy
   dsname = strip(dsname)
   flg = strip(flg)
   if flg = 'XX' | flg ¬= '' then
   do
      say ' Already Processed ' dsname
      iterate
   end
   say  'Processing....' dsname
```

```
   ADDRESS TSO "ALLOCATE DD(INDD2) DSN('"dsname"') REUSE SHR"
   ADDRESS ISPEXEC
       "LMINIT DATAID(DSID) DDNAME(INDD2)"
       "EDIT  DATAID("DSID") MACRO("I_macmem")"
       "LMFREE DATAID(DSID)"
   ADDRESS TSO "FREE  DDNAME(INDD2)"
   flg = '        XX '
   out.1 = dsname||flg
   address tso "execio * DISKW INDD (STEM out. "
   drop out
end
address tso "execio Ø diskw INDD (FINIS "
ADDRESS TSO "FREE DDNAME(INDD)"
exit(Ø)
```

Jaiwant K Jonathan
DB2 DBA
QSS (USA)

DB2HDECP data display

DB2HDECP is a REXX EXEC that displays information from the DSNHDECP ('application defaults') data module.

The EXEC calls an Assembler routine (SWHDECP) that loads the DSNHDECP data module from a DB2 subsystem load library. The SWHDECP module writes a record of the data, and control goes back to the REXX EXEC. The EXEC then formats and displays the data in a pop-up window in the TSO/ISPF environment.

The DSNHDECP data load module is created from the DSNTIJUZ sample job during DB2 installation; and it holds default DB2 application values. IBM has a macro (DSNDDECP) in the SDSNMACS installation library that is used for mapping DSNHDECP.

To invoke the DB2HDECP EXEC, enter 'TSO DB2HDECP xxxx' from a TSO ISPF session. The 'xxxx' is a subsystem ID (eg DB2T, DB2P, etc); if a subsystem ID is not passed, then the EXEC assumes the default is DB2T.

The DSNHDECP data module applicaion programming defaults are shown below:

```
DB2 VERSION--------- DECPREL=  610
DECIMAL ARITH--- DECPAR=   DEC15
PERIOD/COMMA-------- DECPDE=   PERIOD
DECP DYNAMRULES- DECPDRU=  YES
YES/NO MIX GRAPHIC-- DECPGRA=  NO
SCCSID---------- DECPSID=  500
CHARACTER SET------- DECPCHA=  ALPHANUM
MCCSID---------- DECPMID=  65534
STRING DELIMITER---- DECPDL=   DEFAULT
GCCSID---------- DECPGID   65534
SQL STRING DELIMIT-- DECPSDL=  DEFAULT
SQL LANG SUP LV- DECPSQL=  NO
SUBSYSTEM ID-------- DECPSSI=  DB2T
DIST SQL DELIM-- DECPDSD=  APOST
LANGUAGE------------ DECPLAN=  COB2
ASCII SCCSID---- DECPASI=  437
DATE FORMAT--------- DECPDAT=  ISO
ASCII MCCSID---- DECPAMI=  65534
LOCAL DATE LENGTH--- DECPDLE=  0
```

```
ASCII GCCSID---- DECPAGI=  65534
TIME FORMAT--------- DECPTIM=  ISO
ENCODE SCHEME--- DECPENS=  EBCDI
LOCAL TIME LEN------ DECPTLE=  Ø
COMPAT OPTION--- DECPCOM=  OFF
LOCAL--------------- DECPLOC=
```

DB2HDECP.PAN

```
)ATTR
/*-------------------------------------------------------------------*/
/*-------------------------------------------------------------------*/
 @ TYPE(TEXT)    COLOR(BLUE)
 # TYPE(TEXT)    COLOR(TURQ)
 % TYPE(OUTPUT)  COLOR(YELLOW)
)BODY ASIS WINDOW(76,17)

  %SSIDO  @ DSNHDECP DATA MODULE - APPLICATION PROGRAMMING DEFAULTS

@DB2 VERSION---------#DECPREL= %PRELO    @DECIMAL ARITH---#DECPAR=  %PARO
@PERIOD/COMMA--------#DECPDE=  %PDEO     @DECP DYNAMRULES-#DECPDRU= %PDRULO
@YES/NO MIX GRAPHIC--#DECPGRA= %PGRAO    @SCCSID----------#DECPSID= %PSIDO
@CHARACTER SET-------#DECPCHA= %PCHARO   @MCCSID----------#DECPMID= %PMIDO
@STRING DELIMITER----#DECPDL=  %PDLO     @GCCSID----------#DECPGID  %PGIDO
@SQL STRING DELIMIT--#DECPSDL= %PSDLO    @SQL LANG SUP LV-#DECPSQL= %PSQLO
@SUBSYSTEM ID--------#DECPSSI= %PSSIDO   @DIST SQL DELIM--#DECPDSD= %PDSDO
@LANGUAGE------------#DECPLAN= %PLANGO   @ASCII SCCSID----#DECPASI= %PASIDO
@DATE FORMAT---------#DECPDAT= %PDATEO   @ASCII MCCSID----#DECPAMI= %PAMIDO
@LOCAL DATE LENGTH---#DECPDLE= %PDLENO   @ASCII GCCSID----#DECPAGI= %PAGIDO
@TIME FORMAT---------#DECPTIM= %PTIMEO   @ENCODE SCHEME---#DECPENS= %PENSO
@LOCAL TIME LEN------#DECPTLE= %PTLENO   @COMPAT OPTION---#DECPCOM= %PCOMPT
@LOCAL---------------#DECPLOC= %PLOC

)INIT
 &ZWINTL = 'DB2HDECP'
)PROC
)END
```

SWHDECP.PGM

```
*-------------------------------------------------------------------*
* PROGRAM SWHDECP IS CALLED BY A TSO/REXX ROUTINE "TSO DB2HDECP";
* IT IS EXECUTED FROM THE ISPF ENVIRONMENT.
* SWHDECP LOADS DATA MODULE DSNHDECP AND WRITES A RECORD FROM THE DATA.
* REXX EXEC DB2HDECP THEN READS THE RECORD AND FORMATS IT FOR DISPLAY.
*
* THE LOAD MACRO IS USING THE DCB AND LSEARCH=YES PARAMETERS SO THAT
* THE SEARCH WILL BE LIMITED TO JPA AND THE LOAD LIBS TO WHICH THE DCB
```

```
* PARAMETER IS POINTING.  THIS IS BEING DONE SO THAT THE REXX EXEC
* WILL PICK-UP THE DSNHDECP MODULE FROM THE DB2 LOAD LIB 'HDECP' DD
* AND NOT SEARCH THE 'ISPLLIB' DD WHICH MIGHT HAVE ALLOCATED A
* DIFFERENT DB2 LOADLIB IN THE TSO LOGON PROC FROM WHICH THE DB2HDECP
* EXEC HAS BEEN INVOKED.
*-------------------------------------------------------------------*
SWHDECP  CSECT
         TITLE 'DSNHDECP DATA PUT RECORD'
         SPACE 1
RØ       EQU   Ø                       WORK REGISTER
R2       EQU   2                       WORK REGISTER
R11      EQU   11                      CODE/DATA BASE
R12      EQU   12                      CODE BASE
R13      EQU   13                      SAVE AREA
R14      EQU   14                      RET ADDR
R15      EQU   15                      RET CODE
         SPACE 1
         STM   14,12,12(13)            STORE THE REGISTERS
         LR    12,15                   LOAD BASE REGISTER 12
         USING SWHDECP,12              ESTABLISH ADDRESSABILITY
         ST    13,SAVEAREA+4           SAVE CALLER PTR TO REG AREA
         LA    13,SAVEAREA             SET PTR TO OWN REG AREA
         SPACE 1
OPENDEC  OPEN  HDECP                   DD HDECP
OPENFIL  OPEN  (OUTFILE,(OUTPUT))      OPEN OUTPUT FILE
         SPACE 1
*-------------------------------------------------------------------*
*        LOAD DB2 DATA MODULE DSNHDECP
*-------------------------------------------------------------------*
LOADDECP DS    ØH
         LOAD  EP=DSNHDECP,DCB=HDECP,LSEARCH=YES
*                                      LOAD DB2 DATA MOD DSNHDECP.
*                                      LIMIT LOAD MOD SEARCH TO DCB
*                                      AND JPA
         LTR   R15,R15                 CK RTN CDE. IS IT THERE?
         BNZ   LOADERR                 NO? THEN B TO ERRSSID
         LR    R2,RØ                   LOAD ADDR OF DATA BLOCK
         USING DECP,R2                 SET ADDRESSABILITY
         PUT   OUTFILE,DECP            WRITE RECORD FROM DATA MOD MAP
         DROP  R2                      RELEASE R2 AS A BASE REGISTER
         SPACE 1
CLOSEFIL DS    ØH
         CLOSE (OUTFILE)               CLOSE OUTPUT
CLOSEDEC CLOSE HDECP
         SPACE 1
RETNREL  DS    ØH
         L     R15,RETCODE             GET RETURN CODE
         RTN   RC=(15)                 RETURN TO CALLER
         SPACE 1
*-------------------------------------------------------------------*
```

```
*        DSNHDECP LOAD ERROR RTN
*-------------------------------------------------------------------*
LOADERR  DS    ØH
         MVC   RETCODE,=F'16'          SET RETURN CODE TO 16
         B     CLOSEFIL                BRANCH TO CLOSEFIL
*-------------------------------------------------------------------*
*
*-------------------------------------------------------------------*
SAVEAREA DS    18F
RETCODE  DC    F'Ø'
         SPACE 1
*-------------------------------------------------------------------*
*        FILE DEFINITIONS
*-------------------------------------------------------------------*
         SPACE 1
HDECP    DCB   DDNAME=HDECP,           SEARCH THIS DD FOR DSNHDECP     X
               MACRF=R,                                              X
               DSORG=PO
         SPACE 1
OUTFILE  DCB   DDNAME=OUTFILE,         PRINT OUTPUT DEFINITION         X
               MACRF=PM,               DSNHDECM MACRO LTH FOR V5.1     X
               BLKSIZE=26Ø,            IS 94                           X
               LRECL=26Ø,              DSNHDECM MACRO LTH FOR V6.1     X
               DSORG=PS                IS 26Ø
*-------------------------------------------------------------------*
*        MACRO DSNDDECP - DATA MODULE MAPPING MACRO
*-------------------------------------------------------------------*
         DS    ØH
         DSNDDECP                      MACRO RESIDES IN SDSNMACS PDS
ENDDECP  EQU *
         SPACE 1
         END
```

DB2HDECP.REX

```
/* Rexx ------------------------------------------------------------*/
/* Display DSNHDECP data = DB2 application programming defaults.   */
/* ASSEMBLER routine SWHDECP lives in "pgmload"                    */
/* DSNHDECP is loaded from SDC.OPLIB.DB2@.DSNLOAD (where '@' is    */
/* L,T,P,X,Y); the search for the module is limited to the JPA and */
/* the load libs pointed-to by the DCB parm coded in module SWHDECP*/
/* by using the DCB and LSEARCH parms of the LOAD macro            */
/*-----------------------------------------------------------------*/
/* TRACE ?R */

parse upper arg parm

pgmload       = 'SSW.TESTLIB'
db2load_node1 = 'SDC.OPLIB.'
```

```
db2load_node2 = substr(parm,1,4)

/*-------------------------------------------------------------------*/
/* if no parm was passed (or if it's invalid), then default to db2t*/
/*-------------------------------------------------------------------*/
select
   when db2load_node2 = 'DB2T' then
        nop
   when db2load_node2 = 'DB2P' then
        nop
   when db2load_node2 = 'DB2L' then
        nop
   when db2load_node2 = 'DB2Y' then
        nop
   when db2load_node2 = 'DB2X' then
        nop
   when db2load_node2 = 'DB2Z' then
        nop
   otherwise
        db2load_node2 = 'DB2T'
end

db2load_node3 = '.DSNLOAD'
db2load       = db2load_node1≥≥db2load_node2≥≥db2load_node3

/*-------------------------------------------------------------------*/
/* define a libdef for ISPLLIB so that SWHDECP can be found to be  */
/* invoked.  After SWHDECP is called, it will use DD HDECP.        */
/*-------------------------------------------------------------------*/
"ISPEXEC LIBDEF ISPLLIB DATASET ID('"pgmload"','"db2load"')"
"ISPEXEC LIBDEF ISPPLIB DATASET ID('SYS1.ISPF.PANELS')"
"ISPEXEC LIBDEF SYSEXEC DATASET ID('SYS1.ISPF.SYSEXEC')"

ADDRESS tso
pref  = sysvar(syspref)
uid   = sysvar(sysuid)
pgmna = 'SWHDECP'
lnode = 'OUTFILE'
odsn  = pref≥≥.≥≥uid≥≥.≥≥pgmna≥≥.≥≥lnode
"ALLOC FI(OUTFILE) DSN('"odsn"') RECFM(F) NEW DELETE"
"ALLOC FI(HDECP)   DSN('"pgmload"','"db2load"') SHR REUSE"
/*-------------------------------------------------------------------*/
/* Call ASSEMBLER program SWHDECP.  SWHDECP loads data module      */
/* DSNHDECP and puts a record of the data from dsnhdecp to an      */
/* output file.  Upon return from the call to SWHDECP, Rexx EXEC   */
/* DB2HDECP formats the data for display in a pop-up window.       */
/*-------------------------------------------------------------------*/
ADDRESS LINKPGM "SWHDECP"
select
   when RC = Ø then
```

```
      do
         call MAIN_LINE
         ADDRESS tso
         "FREE FI(HDECP,OUTFILE)"
         exit
      end
   when RC = -3 then
      do
         say 'host command environ could not locate pgm swhdecp'
         exit
      end
   otherwise
      do
         say 'return code is ' RC
         exit
      end
end
/*-------------------------------------------------------------------*/
/* Format the data from DSNHDECP and display it in pop-up window   */
/*-------------------------------------------------------------------*/
MAIN_LINE:
   call FORMAT_HDECP
   call POPUP_WINDOW
return
/*-------------------------------------------------------------------*/
/* There is only one input record                                    */
/*-------------------------------------------------------------------*/
FORMAT_HDECP:
ADDRESS tso
newstack
"EXECIO * DISKR OUTFILE (FINIS"

do QUEUED()
   pull record
   prelo   = substr(record,9,4)

   select
      when substr(record,25,1) = '00000000'b then
           pdeo = 'period'
      when substr(record,25,1) = '10000000'b then
           pdeo = 'comma'
      otherwise
           nop
   end
   select
      when substr(record,26,1) = '00000000'b then
           pgrao = 'no'
      when substr(record,26,1) = '10000000'b then
           pgrao = 'yes'
      otherwise
```

```
        nop
end
pcharo  = substr(record,27,8)

select
   when substr(record,35,1) = '01000000'b then /* x'40' */
        pdlo = 'default'
   when substr(record,35,1) = '00000000'b then
        pdlo = 'apost'
   when substr(record,35,1) = '10000000'b then
        pdlo = 'quote'
   otherwise
        nop
end
select
   when substr(record,36,1) = '01000000'b then /* x'40' */
        psdlo = 'default'
   when substr(record,36,1) = '00000000'b then
        psdlo = 'apost'
   when substr(record,36,1) = '10000000'b then
        psdlo = 'quote'
   otherwise
        nop
end

pssido  = substr(record,37,4)
plango  = substr(record,41,8)
pdateo  = substr(record,53,5)
pdleno  = c2d(substr(record,58,1))
ptimeo  = substr(record,59,5)
ptleno  = c2d(substr(record,64,1))

select
   when substr(record,65,1) = '00000000'b then
        paro = 'dec15'
   when substr(record,65,1) = '10000000'b then
        paro = 'dec31'
   otherwise
        nop
end
pdrulo  = c2d(substr(record,66,1))
select
   when substr(record,66,1) = '00000000'b then
        pdrulo = 'no'
   when substr(record,66,1) = '10000000'b then
        pdrulo = 'yes'
   otherwise
        nop
end
```

```
   psido   = c2d(substr(record,67,2))
   pmido   = c2d(substr(record,69,2))
   pgido   = c2d(substr(record,71,2))
   psqlo   = c2d(substr(record,73,1))

   select
      when psqlo = 1 then
           psqlo = 'no'
      when psqlo = 2 then
           psqlo = 'ansi86'
      otherwise
           nop
   end

   pdsdo   = c2d(substr(record,74,1))
   select
      when substr(record,74,1) = '00000000'b then
           pdsdo = 'apost'
      when substr(record,74,1) = '10000000'b then
           pdsdo = 'quote'
      otherwise
           nop
   end

   pasido  = c2d(substr(record,75,2))
   pamido  = c2d(substr(record,77,2))
   pagido  = c2d(substr(record,79,2))
   penso   = c2d(substr(record,81,1))

   select
      when substr(record,81,1) = '00000000'b then
           penso = 'ebcdic'
      when substr(record,81,1) = '01000000'b then /* x'40' */
           penso = 'ascii'
      otherwise
           nop
   end

/*-----------------------------------------------------------------*/
/* The next two fields were added due to DB2 V6.1 changes          */
/*-----------------------------------------------------------------*/
   pcompt  = c2d(substr(record,82,1))
   select
      when substr(record,82,1) = '00000000'b then
           pcompt = 'off'
      when substr(record,82,1) = '10000000'b then
           pcompt = 'on '
      otherwise
           nop
   end
```

```
/*-------------------------------------------------------------------*/
/* The DSNHDECM macro field DECPLCTP has a length of 5Ø, but just  */
/* use the first 8 bytes.                                            */
/*-------------------------------------------------------------------*/
   ploc    = substr(record,83,8)

end
return
/*-------------------------------------------------------------------*/
/* Pop-up window processing                                          */
/*-------------------------------------------------------------------*/
POPUP_WINDOW:
   "ISPEXEC ADDPOP"
   "ISPEXEC DISPLAY PANEL(DB2HDECP)"
   "ISPEXEC REMPOP"
return
```

John Mustric
Senior Systems Programmer
NiSource Corporate Services (USA)

Embarcadero Technologies has announced availability of its DBArtisan database administration tool for DB2 for OS/390, providing Windows-based set-up and operation to manage routine administration tasks. This adds to existing support for Oracle, Microsoft, and Sybase.

Among the product's features are Windows-based client installation, requiring no mainframe software components to get up and running, graphical presentation of current threads, trace activity, running functions, and object locks plus point-and-click automation of error-prone and repetitive tasks.

Also included is shared reporting of DB2 for OS/390 schema definitions using HTML and schema and object management with wizards simplifying schema object creations, modifications, extractions, and migrations.

For further information contact:
Embarcadero Technologies, 425 Market Street, Suite 425, San Francisco, CA 94105, USA.
Tel: (415) 834 3131.
URL:http://www.embarcadero.com/products/administer/db2datasheet.htm.

* * *

ETI has launched its Data System Library (DSL) for C/DB2, which offers the ability to utilize the C language to access DB2 data. This new tool adds to the company's ETIoEXTRACT suite.

DSL for C/DB2 features data transformation capabilities and it also supports the parallel loading process for DB2 UDB.

For further information contact:
ETI, 816 Congress Ave, Suite 1300, Frost Bank Plaza, Austin, TX 78701, USA.
Tel: (512) 383 3000.
URL: http://www.eti.com/press/pr/0101speed.html.

* * *

Serena Software has announced Change Man 5.1, eChange Man 5.3, and ChangeXpress 2.1, designed to provide a single point of control for managing resources and software changes across OS/390, NT, Unix, HP e3000, and OS/400 environments.

Change Man provides the SCM infrastructure for OS/390 MVS applications, while eChange Man supports SCM for Windows, NT, Web, Unix, OS/400, and MPE/iX environments. ChangeXpress provides a single point of control for software change reporting and approvals for platforms supported by Change Man and eChange Man.

Version 5.3 of eChange Man integrates SCM across Windows, Web, and Unix and enhancements include wider client support, support for DB2 UDB databases and Windows 2000, and enhanced build management procedures.

For further information contact:
Serena Software, 500 Airport Blvd, 2nd Floor, Burlingame, CA 94010-1904, USA.
Tel: (650) 696 1800.
URL: http://www.serena.com.