# 119

# DB2

*September 2002*

## In this issue





update

# 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 1999 issue, are available separately to subscribers for £22.50 ($33.75) each including postage.

## *DB2 Update* on-line

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

## 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 download a copy of our *Notes for Contributors* from www.xephon.com/nfc.




*Printed in England.*

## Show all user authorizations

I have written this utility for two reasons – first, to help DBAs with problem solving, and, secondly, for security departments before issuing GRANT/REVOKE statements.

I tested it with OS/390 2.5, TSO 2.6.0, ISPF/PDF 4.5, and DB2 5.2.

This utility brings all user authorizations and privileges from the DB2 catalog for a given userID or groupID. It automatically creates a sequential dataset whose format is <UserID>.SHOAUTH.INF, and writes the information to it.

Part of the output screen is shown in below:

```
Authorizations For USERID: GORO
-------------------------
Table Privileges:
----------------
Grantor  Creator  Table Name          How Alt Del Inx Ins Sel Upd Date
-------- -------- ------------------- -- -- -- -- -- -- -- --------
SØ53Ø    SYSIBM   SYSCOPY             S   -   -   -   -   S   -   Ø2Ø4Ø7
SØ53Ø    SYSIBM   IPNAMES             S   -   -   -   -   S   -   Ø2Ø4Ø7
SØ53Ø    SYSIBM   SYSRESAUTH          S   -   -   -   -   S   -   Ø2Ø4Ø7
GTCDBA   PO       TBOISEX             D   -   -   -   -   S   -   Ø2Ø319
GTCDBA   PO       TBORPRE             D   -   -   -   -   S   -   Ø2Ø319
GTCDBA   PO       TBOSITE             D   -   -   -   -   S   -   Ø2Ø319
```

There are two parts of this utility. The first part is the user interface, which is a combination of REXX, CLIST, and ISPF panel. The second part is a COBOL/DB2 program that gets the information from the DB2 catalog.

The catalog tables used in this utility are:

- SYSIBM.SYSTABAUTH
- SYSIBM.SYSUSERAUTH
- SYSIBM.SYSRESAUTH
- SYSIBM.SYSPLANAUTH
- SYSIBM.SYSPACKAUTH
- SYSIBM.SYSDBAUTH

- SYSIBM.SYSCOLAUTH.

The REXX EXEC is DB2AUTH, the ISPF panel is SHOWAUTH, the CLIST is SHOWAUT2, and the COBOL/DB2 program is SHOWAUTH.

DB2AUTH

```
/*--REXX-------------------------------------------------------*/
/*  Procedure DB2Auth                                         */
/*  Author    Mehmet Cuneyt GOKSU                             */
/*  Date      March   2002                                    */
/*  Purpose   Retrieve Authorizations for a User              */
/*-------------------------------------------------------------*/
  Address ISPEXEC
  If Sysvar(SYSISPF)='NOT ACTIVE' Then Exit
  UserID=''
  Table =''
  DbID='DSNP'
  Authds  = SYSVAR("SYSUID")||'.SHOWAUTH.INF'
  User    = SYSVAR("SYSUID")
      Address TSO
      If User ¬= 'S0530' Then
                     Do
                    "SEND 'DB2AUTH Used By:' USER(S0530) LOGON"
                     End
      Msg_status = MSG('Off')
      "PROFILE NOPREFIX"
      "DELETE "Authds""
      Msg_status = MSG('On')
          "ALLOCATE DA("Authds") NEW REUSE" "DSORG(PS) " ,
          "LRECL(80) RECFM(F) BLKSIZE(80) FI(AUTHDD) " ,
          "STORCLAS(USERSCL)",
          "SPACE("3",1) CYLINDERS "
  Address ISPEXEC
Display_Panel:
  'DISPLAY PANEL(SHOWAUTH)'
  'VGET (ZVERB) SHARED'
  If zverb='END' | zverb='RETURN' Then
    Do
      Address TSO
      "FREE F(AUTHDD)"
      Exit 0
    End
  /*********** Main Loop Starts ****/
  Address Tso
"EXEC 'SYS9.UTILITY.EXECLIB(SHOWAUT2)' '"USERID" "DBID"' CLIST"
"ISPEXEC BROWSE DATASET('"Authds"')"
  Address ISPEXEC
  Signal Display_Panel
  /*********** Main Loop Ends ******/
```

## SHOWAUT2

```
PROC 2 &USERID &DBID
DSN SYSTEM(&STR(&DBID))
RUN PROGRAM (SHOWAUTH) PLAN (SHOWAUTH) -
LIBRARY('PROD.LOADLIB') PARMS('&USERID')
END
EXIT
```

## SHOWAUTH ISPF PANEL

```
)Attr
 @ Type(Output) Caps(Off) Intens(High) Just(Left)
 ¢ Type(Input)  Caps(On)  Intens(High) Just(Left) Pad(_)
)Body Expand(¦¦)
%-------------- SAAOC EDP -- Retrieve DB2 Authorities   %-¦-¦-
%Command ===>_zcmd       ¦ ¦ +
+
+
+
+            Enter User ID or RACF Group
+
+           %UserID    ===>¢UserID  +                              +
+
+
+
+           @MESSAGE                                               +
+
+           %DB2 SysID ===>¢DBID+
+
+
+
+
+-¦-¦-
%PFØ3/15+or%PFØ4/16+- Quit ¦ ¦ %PFØ1/13+- Help
+-¦-¦-
)Init
 .Cursor = UserID
 &Zcmd = &Z
)Proc
 &MESSAGE = &Z
 Refresh(MESSAGE)
 Ver(&UserID,NB,LEN,LE,8)
 Ver(&DBID,NB,LEN,LE,4)
)End
```

## SHOWAUTH COBOL/DB2 PROGRAM

```
       IDENTIFICATION    DIVISION.
       PROGRAM-ID.       SHOWAUTH.
```

```
 DATE-WRITTEN.
 DATE-COMPILED.
*****************************************************************
* FUNCTION:                                                     *
*****************************************************************
*    SHOWAUTH                                                   *
*    Mehmet Cuneyt Goksu  / 2002                                *
*****************************************************************
* PROGRAM ID : SHOWAUTH                                         *
*****************************************************************
 EJECT
 ENVIRONMENT    DIVISION.
 INPUT-OUTPUT    SECTION.
 FILE-CONTROL.
     SELECT  AUTHDD      ASSIGN       TO UT-S-AUTHDD
                         FILE STATUS  IS VSAM-STATUS-2.
 DATA DIVISION.
 FILE SECTION.
*****************************************************************
*          OUTPUTDD                    F I L E                  *
*****************************************************************
 FD  AUTHDD
     BLOCK 0 RECORDS
     RECORDING MODE IS F
     LABEL RECORD STANDARD.
 01  AUTHDD-REC.
   05 AUTH-RECORD                 PIC X(80).
*****************************************************************
*         W O R K I N G   S T O R A G E   S E C T I O N         *
*****************************************************************
 WORKING-STORAGE SECTION.
 01  AUTHDD-REC-WS           PIC X(80) VALUE ALL ' '.
 01  WS-PROG-NAME            PIC  X(27)  VALUE
                             'WORKING STORAGE OF DB2IC  '.
*****************************************************************
*                   VSAM-STATUS CODES COPYBOOK                  *
*****************************************************************
     COPY VSAMAID.
 EJECT
*****************************************************************
*                    M I S C E L L A N E O U S                  *
*****************************************************************
 01  I                           PIC 9(3) VALUE 0.
 01  J                           PIC 9(3) VALUE 0.
 01  K                           PIC 9(1) VALUE 0.
 01  WS-USERID                   PIC X(8) VALUE SPACE.
*****************************************************************
*                        S W I T C H E S                        *
*****************************************************************
 01  SW-SWITCHES.
     05  SW-EOF-TRANS              PIC X.
```

```
         88  EOF-TRANS                    VALUE 'Y'.
         88  NOT-EOF-TRANS                VALUE 'N'.
 Ø1  ERROR-MESSAGE.
         Ø2  ERROR-LEN   PIC S9(4)  COMP VALUE +132Ø.
         Ø2  ERROR-TEXT  PIC X(132) OCCURS 1Ø TIMES
                                    INDEXED BY ERROR-INDEX.
 77  ERROR-TEXT-LEN      PIC S9(9)  COMP VALUE +132.
*************************************************************
*  CURSOR DECLARATIONS                                      *
*************************************************************
     EXEC SQL INCLUDE SQLCA END-EXEC.
     EXEC SQL INCLUDE SYSTABAU END-EXEC.
     EXEC SQL INCLUDE SYSUSRAU END-EXEC.
     EXEC SQL INCLUDE SYSRESAU END-EXEC.
     EXEC SQL INCLUDE SYSPLNAU END-EXEC.
     EXEC SQL INCLUDE SYSPCKAU END-EXEC.
     EXEC SQL INCLUDE SYSDBAUT END-EXEC.
     EXEC SQL INCLUDE SYSCOAUT END-EXEC.
     EXEC SQL DECLARE C1 CURSOR FOR
       SELECT GRANTOR, TCREATOR, TTNAME, AUTHHOWGOT
       , ALTERAUTH, DELETEAUTH, INDEXAUTH, INSERTAUTH
       , SELECTAUTH, UPDATEAUTH, DATEGRANTED
       FROM SYSIBM.SYSTABAUTH
       WHERE GRANTEE = :WS-USERID
       FOR FETCH ONLY
     END-EXEC.

     EXEC SQL DECLARE C2 CURSOR FOR
       SELECT GRANTOR, AUTHHOWGOT
       , BINDADDAUTH, CREATEDBAAUTH, DISPLAYAUTH
       , STOPALLAUTH, SYSADMAUTH, DATEGRANTED
       FROM SYSIBM.SYSUSERAUTH
       WHERE GRANTEE = :WS-USERID
       FOR FETCH ONLY
     END-EXEC.
     EXEC SQL DECLARE C3 CURSOR FOR
       SELECT GRANTOR, AUTHHOWGOT
       , QUALIFIER, NAME, OBTYPE, USEAUTH
       , DATEGRANTED
       FROM SYSIBM.SYSRESAUTH
       WHERE GRANTEE = :WS-USERID
       FOR FETCH ONLY
     END-EXEC.
     EXEC SQL DECLARE C4 CURSOR FOR
       SELECT GRANTOR, AUTHHOWGOT
       , NAME, BINDAUTH, EXECUTEAUTH
       , DATEGRANTED
       FROM SYSIBM.SYSPLANAUTH
       WHERE GRANTEE = :WS-USERID
       FOR FETCH ONLY
     END-EXEC.
```

```
     EXEC SQL DECLARE C5 CURSOR FOR
       SELECT GRANTOR, AUTHHOWGOT
       , COLLID, NAME, BINDAUTH, EXECUTEAUTH
       , COPYAUTH, TIMESTAMP
       FROM SYSIBM.SYSPACKAUTH
       WHERE GRANTEE = :WS-USERID
       FOR FETCH ONLY
     END-EXEC.
     EXEC SQL DECLARE C6 CURSOR FOR
       SELECT GRANTOR, AUTHHOWGOT
       , NAME, CREATETABAUTH, CREATETSAUTH, DBADMAUTH, DBCTRLAUTH
       , DBMAINTAUTH, DISPLAYDBAUTH, DROPAUTH, IMAGCOPYAUTH
       , LOADAUTH, REORGAUTH, RECOVERDBAUTH, REPAIRAUTH
       , STARTDBAUTH, STATSAUTH, STOPAUTH, DATEGRANTED
       FROM SYSIBM.SYSDBAUTH
       WHERE GRANTEE = :WS-USERID
       FOR FETCH ONLY
     END-EXEC.
     EXEC SQL DECLARE C7 CURSOR FOR
       SELECT GRANTOR
       , CREATOR, TNAME, COLNAME, PRIVILEGE, DATEGRANTED
       FROM SYSIBM.SYSCOLAUTH
       WHERE GRANTEE = :WS-USERID
       FOR FETCH ONLY
     END-EXEC.
 LINKAGE SECTION.
 Ø1 WS-DATA.
    Ø5 WS-LL-Z1           PIC S9(4) COMP.
    Ø5 LINK-USERID        PIC X(8) VALUE SPACES.
*************************************************************
 PROCEDURE DIVISION USING WS-DATA.
 ØØØØØØ-PROGRAM-CONTROL.
*****************************************************************
     INSPECT  LINK-USERID REPLACING  ALL  LOW-VALUES
                                          BY  SPACES.
     MOVE LINK-USERID  TO WS-USERID.
     OPEN OUTPUT AUTHDD.
     INSPECT WS-USERID REPLACING ALL LOW-VALUES BY SPACES.
     STRING  'Authorizations For USERID: '  WS-USERID
     DELIMITED BY '  ' INTO AUTH-RECORD.
     PERFORM 99ØØØØ-WRITE-AUTHDD.
     MOVE    '------------------------- ' TO AUTH-RECORD.
     PERFORM 99ØØØØ-WRITE-AUTHDD.
     PERFORM AØØ1-TABLE-AUTH THRU AØØ1-TABLE-AUTH-EXIT.
     PERFORM AØØ1-USER-AUTH  THRU AØØ1-USER-AUTH-EXIT.
     PERFORM AØØ1-USE-AUTH   THRU AØØ1-USE-AUTH-EXIT.
     PERFORM AØØ1-PLAN-AUTH  THRU AØØ1-PLAN-AUTH-EXIT.
     PERFORM AØØ1-PCK-AUTH  THRU AØØ1-PCK-AUTH-EXIT.
     PERFORM AØØ1-DB-AUTH  THRU AØØ1-DB-AUTH-EXIT.
     PERFORM AØØ1-CO-AUTH  THRU AØØ1-CO-AUTH-EXIT.
 ØØØØØØ-EXIT.
```

```
      CLOSE AUTHDD.
      GOBACK.
      EXIT.
***********  COLUMN AUTH **********
  A001-CO-AUTH.
      STRING  'Column Privileges: '
      DELIMITED BY '  ' INTO AUTH-RECORD.
      PERFORM 990000-WRITE-AUTHDD.
      MOVE    '------------------ ' TO AUTH-RECORD.
      PERFORM 990000-WRITE-AUTHDD.
      EXEC SQL OPEN C7 END-EXEC.
      EXEC SQL FETCH C7 INTO
       :WS-CO-GRANTOR
      ,:WS-CO-CREATOR
      ,:WS-CO-TNAME
      ,:WS-CO-COLNAME
      ,:WS-CO-PRIVILEGE
      ,:WS-CO-DATEGRANTED
      END-EXEC.
      IF SQLCODE < 0
       CALL 'DSNTIAR' USING SQLCA ERROR-MESSAGE ERROR-TEXT-LEN
       DISPLAY 'ERROR OCCURED IN FETCH <COLUMN> PRIV.'
       DISPLAY ERROR-MESSAGE
       GO TO 000000-EXIT
      END-IF.
      IF SQLCODE = 100
       STRING  'There is NO column privilege for this user'
       DELIMITED BY '  ' INTO AUTH-RECORD
       PERFORM 990000-WRITE-AUTHDD
       GO TO A001-CO-AUTH-EXIT
      END-IF.
      IF SQLCODE = 0
       STRING 'Grantor  ' 'Creator  ' 'Table Name         '
              'Column Name        ' 'Privilege ' 'Date   '
       DELIMITED BY '                      ' INTO AUTH-RECORD
       PERFORM 990000-WRITE-AUTHDD
       STRING '-------- ' '-------- ' '------------------ '
              '------------------ ' '-------- ' '------ '
       DELIMITED BY '                      ' INTO AUTH-RECORD
       PERFORM 990000-WRITE-AUTHDD
       PERFORM A001-FETCH-C7
         THRU    A001-FETCH-C7-EXIT
         UNTIL SQLCODE NOT EQUAL 0
      END-IF.
      EXEC SQL CLOSE C7 END-EXEC.
  A001-CO-AUTH-EXIT.
        EXIT.
  A001-FETCH-C7.
      IF WS-CO-PRIVILEGE = ' '
       MOVE 'U' TO WS-CO-PRIVILEGE
      ELSE
```

```
        MOVE 'R' TO WS-CO-PRIVILEGE
       END-IF.
       STRING
       WS-CO-GRANTOR ' ' WS-CO-CREATOR ' ' WS-CO-TNAME-TEXT ' '
       WS-CO-COLNAME ' ' WS-CO-PRIVILEGE ' ' WS-DATEGRANTED
       DELIMITED BY '                        ' INTO AUTH-RECORD.
       PERFORM 990000-WRITE-AUTHDD.
      EXEC SQL FETCH C7 INTO
       :WS-CO-GRANTOR
      ,:WS-CO-CREATOR
      ,:WS-CO-TNAME
      ,:WS-CO-COLNAME
      ,:WS-CO-PRIVILEGE
      ,:WS-CO-DATEGRANTED
      END-EXEC.
  A001-FETCH-C7-EXIT.
        EXIT.
***********  DATABASE AUTH **********
  A001-DB-AUTH.
     STRING  '   ' DELIMITED BY '  ' INTO AUTH-RECORD.
     PERFORM 990000-WRITE-AUTHDD.
     STRING  'Database Privileges (Part 1): '
     DELIMITED BY '     ' INTO AUTH-RECORD.
     PERFORM 990000-WRITE-AUTHDD.
     MOVE    '--------------------------- ' TO AUTH-RECORD.
     PERFORM 990000-WRITE-AUTHDD.
     EXEC SQL OPEN C6 END-EXEC.
     EXEC SQL FETCH C6 INTO
      :WS-DB-GRANTOR
     ,:WS-DB-AUTHHOWGOT
     ,:WS-DB-NAME
     ,:WS-DB-CREATETABAUTH
     ,:WS-DB-CREATETSAUTH
     ,:WS-DB-DBADMAUTH
     ,:WS-DB-DBCTRLAUTH
     ,:WS-DB-DBMAINTAUTH
     ,:WS-DB-DISPLAYDBAUTH
     ,:WS-DB-DROPAUTH
     ,:WS-DB-IMAGCOPYAUTH
     ,:WS-DB-LOADAUTH
     ,:WS-DB-REORGAUTH
     ,:WS-DB-RECOVERDBAUTH
     ,:WS-DB-REPAIRAUTH
     ,:WS-DB-STARTDBAUTH
     ,:WS-DB-STATSAUTH
     ,:WS-DB-STOPAUTH
     ,:WS-DB-DATEGRANTED
     END-EXEC.
     IF SQLCODE < 0
      CALL 'DSNTIAR' USING SQLCA ERROR-MESSAGE ERROR-TEXT-LEN
      DISPLAY 'ERROR OCCURED IN FETCH <DATABASE> PRIV.'
```

```
 DISPLAY ERROR-MESSAGE
 GO TO 000000-EXIT
END-IF.
IF SQLCODE = 100
 STRING  'There is NO DATABASE Privilege for this user'
 DELIMITED BY '  ' INTO AUTH-RECORD
 PERFORM 990000-WRITE-AUTHDD
 GO TO A001-DB-AUTH-EXIT
END-IF.
IF SQLCODE = 0
 STRING 'Grantor  '
        'How '
        'Name     '
        'CreTab ' 'CreTs ' 'Dbadm ' 'Dbctrl '
        'Dbmaint ' 'DisDb ' 'Drop ' 'Imag ' 'Date   '
 DELIMITED BY '                      ' INTO AUTH-RECORD
 PERFORM 990000-WRITE-AUTHDD
 STRING '-------- '
        '-- '
        '-------- '
        '------ ' '---- ' '---- ' '------ '
        '------ ' '---- ' '---- ' '---- ' '------ '
 DELIMITED BY '                       ' INTO AUTH-RECORD
```

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

*Mehmet Cuneyt Goksu*
*DB2 Specialist (Turkey)*


## Space calculation for a tablespace

We have developed a program to determine the space allocated and space used for the tablespaces in a database. In our shop, we use this program to determine the space needed to take an image copy of that database. By using this program, we find the currently allocated space for each tablespace and choose a medium to take the back-up. That medium may be a cartridge or a disk. If the back-up is to be taken on disk, then the primary and secondary allocations are calculated using the program.

This program can be used in different ways also:

- To determine whether the space allocations are efficient or not. For example, if a tablespace is allocated as 600 cylinders but only 1

track is used, that tablespace size can be altered.

- To determine whether we have to add more disks to the storage groups.

This program can also be extended to be used for VSAM files.

The program is called from JCL. The first step of the JCL takes an unload of the names of all the tablespaces in a database. In the second step, it uses the names of the tablespaces as input, and writes the name of the tablespace, the space allocated by that tablespace, and the space used by that tablespace for every tablespace in the database.

## THE JCL TO CALL THE REXX PROGRAM

```
//PLISTBAN JOB ,'TABLE UNLOAD',
//   MSGLEVEL=(1,1),MSGCLASS=X,CLASS=4,REGION=6M,TIME=1440
//UNLOADTS EXEC PGM=IKJEFT01
/*************************************************************
/*  This step takes unload of the names of the tablespaces
/*  defined in the database
/*************************************************************
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  DSN SYSTEM(DB0P)
    RUN  PROGRAM(DSNTIAUL) PLAN(DSNTIB61) PARMS('SQL') -
    LIB('DSNDB0P.RUNLIB.LOAD')
  END
/*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD DUMMY
//SYSPUNCH DD DUMMY
/*************************************************************
/*  The unload is taken into a PDS named dba.copy.tsname
/*  For ease in usability the member name is same as the
/*  database name
/*************************************************************
//SYSREC00 DD DSN=DBA.COPY.TSNAME(PDVRM001),DISP=SHR
//SYSIN    DD *
SELECT NAME FROM SYSIBM.SYSTABLESPACE WHERE DBNAME='PDVRM001'
;
//CRESTMT EXEC PGM=IKJEFT01
/*************************************************************
/*  This step calls the REXX program named listcat, from
/*  the dba.copy.exec
/*  The input is the names of the tablespaces in the database
/*  The output is written to a member which has the same name
/*  as the database, in dba.copy.tsname
/*  The REXX program takes the name of the database and the
```

```
/*  vcat name for the database, which is generic for all
/*  the tablespaces
/*************************************************************
//OUTDD    DD   SYSOUT=*
//SYSTSPRT DD   SYSOUT=*
//SYSEXEC  DD   DSN=DBA.COPY.EXEC,DISP=SHR
//UTPRINT  DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//INP      DD   DSN=DBA.COPY.TSNAME(PDVRMØØ1),DISP=SHR
//MERGCAL  DD   DSN=DBA.COPY.MERGCAL(PDVRMØØ1),DISP=SHR
//SYSTSIN   DD  *
%LISTCAT PDVRMØØ1 VRMP
//*
```

## THE REXX PROGRAM

```
/* REXX */
 arg dbname  vcat
/*********************************************************************/
/* The name of the database and the vcat is taken as a parameter     */
/*********************************************************************/
 'execio * diskr inp (stem input. finis'
 'execio Ø diskw mergcal  (finis'
/*********************************************************************/
/*  The input is defined as inp in the calling JCL                   */
/*  The output is defined as mergcal in the calling JCL              */
/*********************************************************************/
 trksize=48 /*The tracksize is given as kilobytes*/
 output.Ø=input.Ø  /*The output lines will be the same as the input  */
 k=1               /* counter                                        */
 do i=1 to input.Ø   /*For every tablespaces in the input file       */
   totalpages=findpages(input.i vcat dbname)
/*calculate the number of pages allocated and used by that tablespace*/
   totalpages=translate(totalpages,' ','-')
   total_allocated=(word(totalpages,1)/1Ø24)/trksize
   total_used=(word(totalpages,2)/1Ø24)/trksize
/* change the rba to number of tracks                               */
   output.k=input.i||' '||format(total_allocated,1Ø,Ø)||' '
   output.k=output.k||format(total_used,1Ø,Ø)
   k=k+1                         /* update counter                  */
 end /* do */
 'execio * diskw mergcal (stem output. '
 exit
findpages: procedure
/*********************************************************************/
/* This procedure listcats the tablespace and calculates the space   */
/* used and allocated. If the tablespace is partitioned, the         */
/* accumulated space allocations are taken                           */
/*********************************************************************/
 arg tsnam vcat dbname
```

```
/*********************************************************************/
/*Tablespace name, vcat and the database name are taken as parameters*/
/*********************************************************************/
 qual=vcat||'.DSNDBC.'||dbname||'.'
 qual=qual||tsnam
 qual=qual||'.i0001.a'
/* The physical dataset name of the tablespace is found             */
/* Since our shop is in version 6, i0001 is used, with Version 7,    */
/* a modification will be needed and the j0001 will have to be       */
/* checked for online reorged tablespaces                            */
 lrc=0       /* Return code s initialized                            */
 z=1         /* This will be used if there are more than one dataset */
 total_allocated_pages=0      /*  Initializations                    */
 total_used_pages=0           /*  Initializations                    */
 do while lrc==0    /* Do while the last return code is 0            */
   d='00'||z        /* a00z dataset will be listcatted               */
   d=right(d,3)     /* That formatting is needed for tablespaces     */
   qual2=qual||d    /* with more that 9 datasets                     */
   z=z+1            /* For the next dataset                          */
   found=0
   x=outtrap(var.)  /* the listcat is taken in to the variable var.  */
   'listcat ENTRIES('''qual2''') ALL'   /* Listcat is taken          */
   lrc=rc           /* return code is taken                          */
   x=outtrap('off') /* Outtrap is closed                             */
   if lrc>0 then    /* If the listcatted dataset exists              */
     found=1
   i=1
   do i=1 to var.0 while found==0
     check=find(var.i,allocation)/* search for allocation in listcat */
     if check=1 then do
       found=1
       j=i+1         /* check next line of allocation               */
       hi_allocated=translate(var.j,' ','-')
       hi_allocated=word(hi_allocated,7)   /* find hi-alloc rba      */
       allocated_pages=hi_allocated
       total_allocated_pages=total_allocated_pages+allocated_pages
       j=i+2        /* check next line for hi-used rba              */
       hi_used=translate(var.j,' ','-')
       hi_used=word(hi_used,7)
       used_pages=hi_used
       total_used_pages=total_used_pages+used_pages
     end
   end
 end

total_pages=total_allocated_pages||'-'||total_used_pages
return total_pages
```

*Banu Bayraktar Ekiz*
*DB2 Systems Programmer*
*Yapi Kredi Telnoloji (Turkey)*

# DB2 OLAP Server with DB2 OLAP Server Miner and DB2 OLAP Server Analyzer

DB2OS VERSION 7.1

DB2 OLAP Server (DB2OS) provides rapid, intuitive, multidimensional analysis for requirements such as 'Show best and worst students in Computer Engineering'. DB2OS includes a no-cost DB2 OLAP Server Miner (DB2OSM) to combine OLAP and data mining technology. The key difference is that OLAP is hypothesis-driven while data mining is discovery-driven. OLAP is best suited for 'what' queries like 'What business units are performing poorly?' or 'What products have cost overruns?'. Data mining is better at 'why' questions used to pinpoint anomalies like 'Why have sales declined in the third quarter in the Northeast?' DB2OS benefits include:

- Integrates Hyperion Essbase OLAP products with DB2 UDB. More than 70 third-party vendors provide analytical tools and applications that interface to the Essbase API. Some of the biggest include Brio, Business Objects, Microsoft, Netscape, and PeopleSoft.
- Combines powerful multidimensional analysis with a comprehensive set of more than 100 built-in financial, mathematical, and statistical functions.
- Provides tools for full-function Web access.
- Supports concurrent read/write operations for 'what-if' applications such as forecasting and budgeting.
- Combines the flexibility of a relational database and performance of an OLAP multidimensional cube. A cube has dimension tables like time, markets, products, measures, and fact tables that contain cube data values. OLAP provides usability tables such as cube catalog table, cube catalog view, cube table, cube view, etc.

For more information, see my OLAP articles in *DB2 Update*: *Using*

*OLAP with DB2,* Issue 102 (April 2001), *DB2 OLAP Server for OS/390 Version 7.1,* Issue 103 (May 2001), and *DB2 OLAP Server for OS/390 V7.1: DB2 Warehouse Manager,* Issue 105 (July 2001).

## DB2OSM

IBM's Almaden Research Center developed DB2 OLAP Server Miner (DB2OSM) as "an opportunity-discovery feature for DB2OS" by marrying discovery analysis with multidimensional analysis. It applies data mining algorithms to OLAP cubes to find *interesting* values, providing analysts with a starting point for deeper exploration.

*Deviation detection* finds interesting values in a loaded OLAP subcube. The subcube is defined by specifying the source cube, number of reportable deviations, and individual dimension members and combinations. Analysts receive identified individual deviations.

For example, to find deviant values in actual sales where Sales is in the Measures dimension and Actual is in the Scenario dimension:

- Extract OLAP data corresponding to Sales and Actual dimensions.
- Apply the deviation detection mining algorithm to extracted cells.
- Write all deviations to OLAP Miner Deviation Viewer or a spreadsheet providing deviation *magnitude.* Magnitude shows the calculated difference from surrounding values with larger magnitudes being more significant.
- Analysts can use Deviation Viewer or DB2OS spreadsheet to continue investigation of specific red highlighted deviant cells.

## PATENT

IBM designed and optimized the deviation detection algorithm to apply statistical models and data mining techniques to OLAP subcubes to find deviant values. IBM believes deviation detection is an 'invention' worthy of a patent. The title is *A method for determination of an exception in multi-dimensional data*. It contains substantive information on how deviation detection works, including how to:

- Determine an expected value for a set of cells in an OLAP subcube using ANOVA (ANalysis Of VAriance between groups). ANOVA's major benefit is having one result for normal and null hypotheses.
- Determine each cell residue.
- Scale residuals for magnitude.
- Compare scaled residues to a threshold value to isolate deviant values.

IBM's goal is to provide an easy method for ensuring that no abnormal data patterns are missed when exploring a data cube of any data dimensions or hierarchies. Paradigms of pre-excavated exceptions are used to drastically reduce the quantity of drill-downs and selection steps normally required by large data cubes.

PROCEDURE

The general procedure is:

- Calculate expected values for all cells *i* of an OLAP cube using ANOVA.
- Residual *i* = observed value of cell *i* minus expected value of cell *i*.
- Calculate standard deviation for all cells *i* of the same aggregate value (standardized residual *i* = residual *i* divided by standard deviation).
- Determine whether standardized residual *i* > threshold.

DB2OSA

DB2 OLAP Server Analyzer (DB2OSA) is an add-on feature allowing end users to access analytical data in the enterprise or Web. It employs point-and-click, colour-coding, exception alerts, calculated metrics, etc, so novices and experts can do complex analyses.

Developers use the Analyzer API Toolkit (DB2OSAT) to invoke DB2OSA functions in Web-based applications using standard

development tools like Java or HTML. Use Java for highly interactive data navigation and manipulation. Employ HTML for thin-client deployment to many users. Use Java API to develop fully customized applications.

SUMMARY

DB2OSM adds another powerful analytical tool for effectively exploring and mining OLAP cubes. DB2OSA and DB2OSAT add Web interfacing and access tools making DB2OS a complete package allowing IBM to get closer to its goal of providing a single version of truth at all organizational levels.

Caveat: the data must be accurate and current. The old DP maxim of Garbage In, Garbage Out (GIGO) still applies, particularly when so many users are making important decisions based on that data.

REFERENCES

M Keller & D Roller, *A method for determination of an exception in multi-dimensional data*, IBM. Available from IBM.

The author recommends acquiring *The IBM Business Intelligence Solutions* CD for extensive examples including Return on Investment (ROI). The CD also contains many white papers. Available from IBM.

*Eric Garrigue Vesely*
*Principal/Analyst*
*Workbench Consulting (Malaysia)*



## DB2 Recovery Log – detailed analysis of user update activities

Our computer system serves a large number of customers, so online transactions (triggered from a wide network of classic terminals, Internet, and devices like answering machines and ATMs), are processed round-the-clock, and large-scale batch work is carried out as well. In this environment there are times when the need arises to find out by

 

whom and when some data was changed. To satisfy these requests we developed an online auditing tool for tracing the history of changes made to user DB2 tables.

Extracting data from a recovery log is not an easy task because considerable work has to be done to locate the log range where information resides and, afterwards, the detailed report produced by the DSN1LOGP utility is not particularly readable. So the following procedure is designed to filter criteria like dates, times, and tables, interpret the log records that represent user record changes (insert, update, and delete records in a data page), and display the output in a convenient presentation form.

Our installation is currently using DB2 OS/390 Version 5 with production and test subsystems installed. The procedure consists of:

- LOGR0 – application start-up REXX EXEC.
- LOGR1 – log information (main REXX EXEC)
- LOGP0 – log information (main input panel)
- LOGP1 – log information (user tables display panel)
- LOGP2 – log information (log summary display panel).

The main REXX EXEC calls the stand-alone utilities DSN1PRNT, DSNJU004, and DSN1LOGP and the program SQLISPF (published in the July and August 2001 issues of *DB2 Update*). Regarding the fact that the DSN1LOGP utility cannot be used on active log datasets when DB2 is running, if the change being considered has not already been archived, the command -ARCHIVE LOG (which initiates the off-load task) should be executed. It is also assumed that archive logs are stored on tape.

The parameters required on the main input panel are DB2 subsystem name (DSN or DBT at our installation) and the date (given in the form yyyy/mm/dd) when the modification occurred. The names of the DB2 load modules library, bootstrap, and Syslgrnx VSAM dataset are generated according to the chosen DB2 subsystem. The main input panel is shown below:

```
---------------------- LOG INFO - MAIN ---------------------------
```

```
DB2 subsystem     ===> DSN

Date From:        ===> 2001/07/24
Date To:          ===> 2001/07/24

DB2 load library  ===> DSN510.SDSNLOAD
Bootstrap dataset ===> DSNC510.BSDS01
SYSLGRNX dataset  ===> DSNC510.DSNDBC.DSNDB01.SYSLGRNX.I0001.A001
```

The next panel, which is shown below, lists all the user tables. The list can be searched by database, table space, or table name, and then the required table (and the corresponding tablespace) can be selected.

```
------------- LOG INFO - USER TABLES DISPLAY - Row 332 to 345 of 376

Command ===>                                          Scroll ===> CSR
(Search: F object value     object: dbname\tsname\tbname)

DB2 subsystem ==> DSN    Date From: 2001/07/24   Date To: 2001/07/24
Line cmds: S-Select
-----------------------------------------------------
S  DBNAME   TSNAME   TBNAME            DBID PSID OBID
-----------------------------------------------------
S  DBTERDP  TSTERD05 TBTERD14          0115 0072 0056
   DBTERDP  TSTERD06 TBTERD20          0115 0074 006F
   DBTERDP  TSTERD07 TBTERD21          0115 0076 0079
   DBTERDP  TSTERD08 TBTERD22          0115 0066 0067
   DBTERDP  TSTERD09 TBTERD23          0115 006B 006C
   DBTMISP  TMISS001 IZVOR_SRED        0116 0002 0010
   DBTMISP  TMISS001 KRED_ZAH          0116 0002 0012
```

Log ranges relating to the selected table space and the previously chosen date frame are extracted from the output of the DSN1PRNT utility run on the SYSLGRNX table space. Then, by using the list of archive log datasets obtained from the DSNJU004 utility and taking into account only those where extracted log ranges belong, we reduce the amount of the log that, after being copied to a temporary dataset, will be processed further.

Below is a modified DSN1LOGP summary report for the selected table space.The last two columns in the panel represent the disposition and status of the unit of recovery (UR), while the meaning of other columns is obvious. Focusing on the concrete UR log range is pretty easy, because we usually have some additional information, like approximate time of the modification, connection-name, correlation-id, authorization ID, and/or plan used to update the specific table.

```
---------------- LOG INFO - LOG SUMMARY DISPLAY - Row 1 to 9 of 107,258

 Command ===>                                              Scroll ===> CSR
 (Search: F object value     object: connid\corrid\authid\plan\startlog)

 DB2 subsystem ==> DSN     Date From: 2001/07/24
                             Date To: 2001/07/24
                              DBNAME: DBTERDP
                              TSNAME: TSTERD05
                              TBNAME: TBTERD14

 Line cmds: S-Select
-------------------------------------------------------------------------
S CONNID  CORRID   AUTHID  PLAN     DATE   TIME  STARTLOG     ENDLOG       D S
-------------------------------------------------------------------------
  BATCH   TRACDK06 OBRJO03 TRAC6241 01.205 01:46 001AA52B945C 001AA69AB4FB C C
  BATCH   TRACDU04 OBRJO03 TRAC5282 01.205 04:46 001AA69D10C4 001AA69DCC8C C C
  PSCICS2 ENTRTRAP IBMUSER TRAC5920 01.205 06:16 001AA80E13D8 001AA80E1AD2 C C
S PSCICS2 ENTRTRAP IBMUSER TRAC5920 01.205 06:23 001AA80E4947 001AA80E4BED C C
  PSCICS2 ENTRTRAP IBMUSER TRAC5920 01.205 06:26 001AA80EB90A 001AA80EBBB0 C C
  PSCICS2 ENTRTRAP IBMUSER TRAC5920 01.205 06:26 001AA80EBE31 001AA80EC153 C C
```

Then, based on output from the DSN1LOGP utility (this time limited to log records that represent data changes for the just selected log range) and extracting records for the selected table only, the final, easy-to-analyse, detailed report is generated as shown below:

```
   Menu  Utilities  Compilers  Help
******************************************************************************
 BROWSE    SYSADM.LOGOUT.PRIV                 Line 00000000 Col 001 080
****************************** Top of Data ******************************
DB2 subsystem: DSN
       DBNAME: DBTERDP
       TSNAME: TSTERD05
       TBNAME: TERDP.TBTERD14

STARTLOG: 001AA80E4947
  ENDLOG: 001AA80E4BED

CONNID   CORRID   AUTHID   PLAN     DATE   TIME  DISP STATUS
======== ======== ======== ======== ====== ===== ==== ======
PSCICS29 ENTRTRAP IBMUSER  TRAC5920 01.205 06:23 C    C

***************** Begin-of-detailed-report ****************

INSERT IN A DATA PAGE

   ACCOUNT            : 6257275
   VALDATE            : 24.07.2001
   CHANGETYPE         : 1
```

```
   DOCTYPE            : 502
   AMMOUNT            : 100000
   CHECKNUM           : 0
   DOCNUM             : 0
   TIMESTMP           : 2000-07-24-10.10.10.880502
   COMM               : 1.1234567E+02

 *** End-of-detailed-report ***
*************************** Bottom of Data ***************************
```

Several excerpts from different reports are shown below, just to present some cases of various data formats. This procedure covers the following column data types: INTEGER, SMALLINT, DECIMAL, FLOAT, REAL, CHARACTER, VARCHAR, DATE, TIME, and TIMESTAMP. VARCHAR is supported in all kinds of data change if, as recommended, it represents the data type of the last column(s) in the table.

```
UPDATE NOT IN-PLACE , DATA PART ONLY IN A DATA PAGE

   TIMESTMP
         REDO: 2101-03-14-12.47.10.191459
         UNDO: 2000-07-24-10.10.10.880502
   COMM
         REDO: 1.1234567E+02
         UNDO: 1.1234567E+02

   ..............................

UPDATE IN-PLACE IN A DATA PAGE

   AMOUNT
         REDO: 218019
         UNDO: 100000
```

SOURCE CODE

## LOGR0

```
/* rexx LOGR0 */
address ispexec 'select panel(logp0)'
```

## LOGP0

```
)ATTR
%  TYPE(TEXT)
[  TYPE(TEXT) INTENS(LOW)
<  TYPE(INPUT) CAPS(ON)
+  TYPE(TEXT) INTENS(LOW)
```

```
!  TYPE(OUTPUT) INTENS(LOW) CAPS(OFF)
)BODY DEFAULT(]*;)EXPAND($$)
%-$-$- LOG INFO - MAIN -$-$-[

+DB2 subsystem     ===><Z   [
[

+Date From:        ===><Z         [
[
+Date To:          ===><Z         [
[

+DB2 load library  ===><Z                                  [
+Bootstrap dataset ===><Z                                  [
+SYSLGRNX dataset  ===><Z                                              [









<Z[

)INIT
  .ZVARS = '(DSN8SSID DATEFROM DATETO db2load bsds1 lgrnx ZCMD)'
  .CURSOR = DSN8SSID
  &DATEFROM = &ZDATESTD
  &DATETO = &ZDATESTD
  VGET (DSN8SSID DATEFROM DATETO db2load bsds1) SHARED
  IF (&DSN8SSID = &Z)
    &DSN8SSID = DSN
  IF (&db2load = &Z)
    &db2load = DSN510.SDSNLOAD
  IF (&bsds1 = &Z)
    &bsds1 = DSNC510.BSDS01
  IF (&lgrnx = &Z)
    &lgrnx = DSNC510.DSNDBC.DSNDB01.SYSLGRNX.I0001.A001
)PROC
  IF (&DSN8SSID = DSN)
    &bsds1 = DSNC510.BSDS01
    &lgrnx = DSNC510.DSNDBC.DSNDB01.SYSLGRNX.I0001.A001
  IF (&DSN8SSID = DBT)
    &bsds1 = DBTC510.BSDS01
    &lgrnx = DBTC510.DSNDBC.DSNDB01.SYSLGRNX.I0001.A001
  VER  (&DSN8SSID,NB,LIST,DSN,DBT)
```

```
  VER  (&DATEFROM,PICT,'9999/99/99')
  VER  (&DATETO,PICT,'9999/99/99')
  VER (&db2load,NONBLANK)
  VER (&bsds1,NONBLANK)
  VER (&lgrnx,NONBLANK)
  VPUT (DSN8SSID DATEFROM DATETO db2load bsds1 lgrnx) SHARED
  &ZSEL = TRANS(TRUNC(&ZCMD,'.')
               ' ','CMD(LOGR1)'
               '*','?')
)END
```

## LOGP1

```
)ATTR
%  TYPE(TEXT)
[  TYPE(TEXT) INTENS(LOW)
<  TYPE(INPUT) CAPS(ON)
+  TYPE(TEXT) INTENS(LOW)
]  TYPE(OUTPUT) INTENS(LOW) CAPS(OFF)
!  TYPE(INPUT) INTENS(LOW) PADC(' ') CAPS(ON)
*  TYPE(OUTPUT) INTENS(LOW) COLOR(PINK)
;  TYPE(OUTPUT) INTENS(LOW) COLOR(TURQ)
^  TYPE(OUTPUT) INTENS(LOW) COLOR(YELLOW)
)BODY DEFAULT(/,_)EXPAND($$)
%-$-$- LOG INFO - USER TABLES DISPLAY -$-$-[

%Command ===><Z                                                [%Scroll
===><Z   [
 (Search:%F object value[    object: dbname\tsname\tbname)

+DB2 subsystem ==>]Z   [ +Date From:]Z         [ +Date To:]Z         [
%Line cmds:+S-Select [
+-------------------------------------------------[
%S  DBNAME[% TSNAME[% TBNAME         [  DBID PSID OBID
+-------------------------------------------------[
)MODEL
!Z *Z       ;Z       ^Z               ]Z   ]Z   ]Z   [
)INIT
   .ZVARS = '(ZCMD     ZSCR     DSN8SSID DATEFROM DATETO   OPT +
              TDBNAME  TTSNAME  TTBNAME  TDBID    TTSID    TTBID  )'
)END
```

## LOGP2

```
)ATTR
%  TYPE(TEXT)
[  TYPE(TEXT) INTENS(LOW)
<  TYPE(INPUT) CAPS(ON)
+  TYPE(TEXT) INTENS(LOW)
]  TYPE(OUTPUT) INTENS(LOW) CAPS(OFF)
!  TYPE(INPUT) INTENS(LOW) PADC(' ') CAPS(ON)
```

```
*  TYPE(OUTPUT) INTENS(LOW) COLOR(PINK)
;  TYPE(OUTPUT) INTENS(LOW) COLOR(TURQ)
^  TYPE(OUTPUT) INTENS(LOW) COLOR(YELLOW)
)BODY DEFAULT(/,_)EXPAND($$)
%-$-$- LOG INFO - LOG SUMMARY DISPLAY -$-$-[

%Command ===><Z                                                [%Scroll
===><Z   [
 (Search:%F object value[    object: connid\corrid\authid\plan\startlog)

+DB2 subsystem ==>]Z   [  +Date From:]Z         [
                            +Date To:]Z         [
                             +DBNAME:]Z       [
                             +TSNAME:]Z       [
                             +TBNAME:]Z                [

%Line cmds:+S-Select [
+-------------------------------------------------------------------------
-------[
%S CONNID  CORRID   AUTHID  PLAN     DATE   TIME  STARTLOG     ENDLOG
D S[
+-------------------------------------------------------------------------
-------[
)MODEL
!Z]Z      ]Z       ^Z      *Z       ]Z     ]Z    ]Z           ]Z
]Z]Z[
)INIT
   .ZVARS = '(ZCMD ZSCR DSN8SSID DATEFROM DATETO DBNAME TSNAME TBNAME +
              OPT CONNID CORRID AUTHID PLAN DATE TIME STARTLOG ENDLOG +
              DISPOSIT STATUS)'
)END
```

## LOGR1

```
/* REXX - LOGR1 ********************************************************/
/* TITLE    :  LOG INFO                                                */
/* ******************************************************************* */
/* TRACE I                                                             */
  address ISPEXEC
  'CONTROL ERRORS RETURN '
  'VGET (DSN8SSID DATEFROM DATETO db2load bsds1 lgrnx) SHARED'
  userid   = userid()
  tick     = ''''
  DB2V = DSN8SSID
  panel = 'LOGP1'
  tbnam = 'LOGT1'
  tvars = 'TDBNAME TTSNAME TCREATOR TTBNAME TDBID TTSID TTBID'
  cnt_f = Ø
  cnt_ft = ' '
  msg  = ' '
  csrrow = 1
```

```
cursor  = 'OPT'
ADDRESS ISPEXEC ,
'TBERASE 'tbnam
'TBOPEN 'tbnam' WRITE SHARE '
if rc > Ø then do
  'TBCREATE 'tbnam' NAMES('tvars') NOWRITE SHARE '
  if rc ^= Ø then call sql_error rc
  SQLQUERY = "SELECT A.DBNAME, ",
                     "A.TSNAME, ",
                     "A.CREATOR, ",
                     "A.NAME AS TBNAME, ",
                     "HEX(A.DBID) AS DBID, ",
                     "HEX(B.PSID) AS TSID, ",
                     "HEX(A.OBID) AS TBID ",
             "FROM SYSIBM.SYSTABLES A, SYSIBM.SYSTABLESPACE B ",
             "WHERE A.TYPE = 'T' AND ",
                   "A.CREATOR <> 'SYSIBM' AND ",
                   "A.DBNAME NOT LIKE 'DSN%' AND ",
                   "A.DBNAME NOT LIKE 'DSQ%' AND ",
                   "A.DBNAME NOT LIKE 'ADB%' AND ",
                   "A.DBNAME NOT LIKE 'RAA%' AND ",
                   "A.DBNAME NOT LIKE 'RDB%' AND ",
                   "A.DBNAME NOT IN ('CELDIAL', 'DRLDB', 'DBEDB1') ",
               "AND A.TSNAME = B.NAME AND ",
                   "A.DBNAME = B.DBNAME ",
             "ORDER BY 1, 2, 4";
  ADDRESS ISPEXEC "SELECT PGM(SQLISPF)";
  if  _nrows = Ø then do
    zedlmsg = 'No record found'
    'setmsg msg(ISRZØØØ) '
    exit 2Ø
  end
  do i = 1 to _nrows
    TDBNAME         = strip(value(_vn.1"."strip(i,l,'Ø')))
    TTSNAME         = strip(value(_vn.2"."strip(i,l,'Ø')))
    TCREATOR        = strip(value(_vn.3"."strip(i,l,'Ø')))
    TTBNAME         = strip(value(_vn.4"."strip(i,l,'Ø')))
    TDBID           = strip(value(_vn.5"."strip(i,l,'Ø')))
    TTSID           = strip(value(_vn.6"."strip(i,l,'Ø')))
    TTBID           = strip(value(_vn.7"."strip(i,l,'Ø')))
    'TBADD 'tbnam''
  end
  'TBTOP 'tbnam
end
disprc = Ø
do while (disprc < 8)
  opt = ' '
  'TBQUERY 'tbnam' ROWNUM(rowcnt)'
  if csrrow <= Ø then csrrow = 1
  'TBDISPL 'tbnam' PANEL('panel') CSRROW('csrrow') MSG('msg') ,
           CURSOR('cursor') AUTOSEL(NO)'
```

```
    disprc = rc
    if disprc < 8 then do
      if word(strip(ZCMD), 1) = 'F' &,
        (word(strip(ZCMD), 2) = 'DBNAME' |,
         word(strip(ZCMD), 2) = 'TSNAME' |,
         word(strip(ZCMD), 2) = 'TBNAME') then do
        if cnt_ft ¬= word(strip(ZCMD), 3) then do
          cnt_ft = word(strip(ZCMD), 3)
          cnt_f  = 0
        end
        else cnt_f = cnt_f + 1
        call tab_search tbnam 'T'
      end
      else do
        if opt = 'S' then call process_s   /* Process line command */
        disprc = 0
      end
    end
  end
  'TBCLOSE 'tbnam
EXIT

/* *** PROCEDURES ************************************************** */

sql_error:
  parse arg errcode
  zedlmsg = 'Error 'errcode
  'setmsg msg(ISRZ000) '
  exit 20
return

tab_search:
  parse arg tbnamp prefix
  'TBVCLEAR 'tbnamp
  tab_field = prefix || word(strip(ZCMD), 2)
  obj = prefix || word(strip(ZCMD), 2)
  interpret obj ' = word(strip(ZCMD), 3)'
  if cnt_f > 0 then 'TBSKIP 'tbnamp' NUMBER('csrrow') NOREAD'
  'TBSARG 'tbnamp' NEXT NAMECOND('tab_field',EQ)'
  'TBSCAN 'tbnamp' NOREAD POSITION('crpname')'
  csrrow = crpname
  drop crpname
  if rc ¬= 0 then do
    zedlmsg = 'Not found. '
    'setmsg msg(ISRZ000) '
  end
return

  process_s:
    DBNAME = TDBNAME
    TSNAME = TTSNAME
```

```
TBNAME = TTBNAME
SQLQUERY = "SELECT NAME, ",
                  "COLNO, ",
                  "COLTYPE, ",
                  "LENGTH, ",
                  "SCALE, ",
                  "NULLS, ",
                  "KEYSEQ ",
           "FROM SYSIBM.SYSCOLUMNS ",
           "WHERE TBNAME = '" ||TBNAME||"' AND ",
                 "TBCREATOR = '"||TCREATOR||"' ",
           "ORDER BY 2";
ADDRESS ISPEXEC "SELECT PGM(SQLISPF)";
if  _nrows = 0 then do
  zedlmsg = 'No record found'
  'setmsg msg(ISRZ000) '
  exit 20
end
do i = 1 to _nrows
  NAME.i    = strip(value(_vn.1"."strip(i,l,'0')))
  COLNO.i   = strip(value(_vn.2"."strip(i,l,'0')))
  COLTYPE.i = strip(value(_vn.3"."strip(i,l,'0')))
  LENGTH.i  = strip(value(_vn.4"."strip(i,l,'0')))
  SCALE.i   = strip(value(_vn.5"."strip(i,l,'0')))
  NULLS.i   = strip(value(_vn.6"."strip(i,l,'0')))
  KEYSEQ.i  = strip(value(_vn.7"."strip(i,l,'0')))
end
ltab = i - 1
'CONTROL DISPLAY SAVE    '
panel1  = 'LOGP2'
tbnam1  = 'LOGT2'
tvars1  = 'CONNID CORRID AUTHID PLAN DATE TIME STARTLOG ENDLOG'||,
          ' DISPOSIT STATUS'
cnt_f  = 0
cnt_ft = ' '
msg     = ' '
csrrow = 1
cursor = 'OPT'
ADDRESS ISPEXEC ,
'TBERASE 'tbnam1
'TBOPEN 'tbnam1' WRITE SHARE'
if rc > 0 then do
  'TBCREATE 'tbnam1' NAMES('tvars1') NOWRITE SHARE '
  if rc ¬= 0 then call sql_error rc
  call process_s_det
  'TBTOP 'tbnam1
end
disprc = 0
do while (disprc < 8)
  opt = ' '
  'TBQUERY 'tbnam1' ROWNUM(rowcnt)'
```

```
    if csrrow <= Ø then csrrow = 1
    'TBDISPL 'tbnam1' PANEL('panel1') CSRROW('csrrow') MSG('msg') ,
             CURSOR('cursor') AUTOSEL(NO)'
    disprc = rc
    if disprc < 8 then do
      if word(strip(ZCMD), 1) = 'F' &,
        (word(strip(ZCMD), 2) = 'CONNID' |,
         word(strip(ZCMD), 2) = 'CORRID' |,
         word(strip(ZCMD), 2) = 'AUTHID' |,
         word(strip(ZCMD), 2) = 'PLAN' |,
         word(strip(ZCMD), 2) = 'STARTLOG') then do
        if cnt_ft ¬= word(strip(ZCMD), 3) then do
          cnt_ft = word(strip(ZCMD), 3)
          cnt_f  = Ø
        end
        else cnt_f = cnt_f + 1
        call tab_search tbnam1 ''
      end
      else do
        if opt = 'S' then call process_s1  /* Process line command */
      end
    end
  end
  address tso "delete "arch1
  'TBCLOSE 'tbnam1
  'CONTROL DISPLAY RESTORE '
return

process_s_det:
  signal off error
  icnt = Ø
  yyyy = substr(DATEFROM, 1, 4)
  mm   = substr(DATEFROM, 6, 2)
  dd   = substr(DATEFROM, 9, 2)
  DATEFROM1 = yyyy||mm||dd
  yyyy = substr(DATETO, 1, 4)
  mm   = substr(DATETO, 6, 2)
  dd   = substr(DATETO, 9, 2)
  DATETO1 = yyyy||mm||dd
  NUMERIC DIGITS 5Ø
  printlib = tick||userid||'.LOGFØ.PRIV'||tick
  address tso "free fi(sysprint)"
  if sysdsn(printlib) = 'OK' then
    address tso "alloc fi(sysprint) da("printlib") shr "
  else
    address tso "alloc fi(sysprint) da("printlib") new ",
                "  dsorg(ps) space(4ØØØ,1ØØØ) tracks release",
                "  lrecl(133) blksize(2793Ø) recfm(f b a)"
  address tso "alloc fi(sysut1) da('"lgrnx"') shr "
  signal on error
  say 'Execute DSN1PRNT utility'
```

```
address tso "CALL '"||db2load||"(DSN1PRNT)' 'PRINT,FORMAT'"
signal off error
address tso "free fi(sysprint)"
address tso "free fi(sysut1)"
indsn = tick||userid||'.LOGFØ.PRIV'||tick
address tso "free  fi(indd)"
address tso "alloc fi(indd) da("indsn") shr"
address tso "delstack"
stemi = Ø
do forever
  address tso "execio 1 diskr indd"
  if rc = 2 then do
    signal eof_rtn
  end
  pull inrec
  if index(inrec,'DSN1994I') > Ø then do
    leave
  end
  if index(inrec,'RECORD:') > Ø then do
    address tso "execio 1 diskr indd"
    if rc = 2 then do
      signal eof_rtn
    end
    pull inrec
    DBIDTSID = TDBID || TTSID
    if index(inrec, DBIDTSID) > Ø then do
      parse var inrec,
        12 cmm1 13 ,
        14 cmm2 15 ,
        16 cdd1 17 ,
        18 cdd2 19 ,
        21 cgg1 22 ,
        23 cgg2 24 ,
        42 csrba1 46 ,
        47 csrba2 55 ,
        56 cerba1 64 ,
        65 cerba2 69
      address tso "execio 1 diskr indd"
      if rc = 2 then do
        signal eof_rtn
      end
      pull inrec
      parse var inrec,
        Ø2 cslrs1 1Ø ,
        11 cslrs2 15 ,
        15 celrs1 19 ,
        2Ø celrs2 28
      if cgg1 = Ø then do
        curryyyymmdd = "2Ø"
      end
      else do
```

```
            curryyyymmdd = "19"
          end
          curryyyymmdd = curryyyymmdd || ,
                           cgg1 || cgg2 || cmm1 || cmm2 || cdd1 || cdd2
          if (curryyyymmdd >= DATEFROM1) & ,
             (curryyyymmdd <= DATETO1) then do
            stemi = stemi + 1
            currstartrba.stemi = csrba1 || csrba2
            currendrba.stemi   = cerba1 || cerba2
            currstartlrs.stemi = cslrs1 || cslrs2
            currendlrs.stemi   = celrs1 || celrs2
          end
        end
      end
    end   /* end do forever */
eof_rtn:
    address tso "free fi(sysprint)"
    address tso "free fi(sysut1)"
    address tso "execio Ø diskr indd(finis"
    address tso "delete "indsn
    printlib = tick||userid||'.LOGF1.PRIV'||tick
    address tso "free fi(sysut1)"
    address tso "free fi(sysprint)"
    address tso "alloc fi(sysut1) da('"bsds1"') shr"
    if sysdsn(printlib) = 'OK' then
      address tso "alloc fi(sysprint) da("printlib") shr "
    else
      address tso "alloc fi(sysprint) da("printlib") new ",
                  "  dsorg(ps) space(4,2) tracks",
                  "  lrecl(125) blksize(6144) recfm(v b a)"
    signal on error
    say 'Press ENTER to list BSDS'
    address tso "CALL '"||db2load||"(DSNJU004)'"
    signal off error
    arch1   = tick||userid||'.LOGF2.PRIV'||tick
    if sysdsn(arch1) = 'OK' then
      address tso "alloc fi(archive) da("arch1") shr "
    else
      address tso "alloc fi(archive) da("arch1") new ",
                  "  dsorg(ps) space(2000,1000) tracks",
                  "  lrecl(4096) blksize(24576) recfm(f b)"
    address tso "free fi(sysin)"
    sysi  = tick||userid||'.SYSIN.PRIV'||tick
    if sysdsn(sysi) = 'OK' then do
      address tso "alloc fi(sysin) da("sysi") shr "
    end
    else
      address tso "alloc fi(sysin) da("sysi") new ",
                  "  dsorg(ps) space(1,1) tracks",
                  "  lrecl(80) blksize(27920) recfm(f b)"
    address tso "delstack"
```

```
queue  '  OPTION COPY'
address tso "execio 1 diskw sysin"
address tso "execio Ø diskw sysin(finis"
address tso "free  fi(sysout)"
address tso "alloc fi(sysout) dummy "
indsn   = tick||userid||'.LOGF1.PRIV'||tick
address tso "free  fi(indd)"
address tso "alloc fi(indd) da("indsn") shr"
address tso "delstack"
do forever
  address tso "execio 1 diskr indd"
  if rc = 2 then do
    signal eof_rtn
  end
  pull inrec
  if index(inrec,'ARCHIVE LOG COPY 1') > Ø then do
    leave
  end
end
do forever
  address tso "execio 1 diskr indd"
  if rc = 2 then do
    signal eof_rtn
  end
  pull inrec
  if index(inrec,'ACTIVE LOG') > Ø then do
    leave
  end
  if index(inrec,'DSN=') > Ø then do
    parse var inrec,
      Ø5 currentstartrba 17 ,
      28 currentendrba 4Ø ,
      68 currentlogdsn 112
    iterate
  end
  if index(inrec,'UNIT=') > Ø then do
    parse var inrec,
      Ø3 currentstartyyddd 11 ,
      84 currentvolume 9Ø
  end
  else do
    iterate
  end
  if currentstartyyddd = 'ØØ.ØØØ' then do
    iterate
  end
  fnd = Ø
  do stemj = 1 to stemi
    if (X2D(currstartrba.stemj) >= ,
        X2D(currentstartrba)) & ,
       (X2D(currstartrba.stemj) <= ,
```

```
        X2D(currentendrba)) then do
      fnd = 1
      leave
    end
  end
  if fnd = 1 then do
    icnt = icnt + 1
    if icnt = 1 then do
      rbastartx = currentstartrba
      arch = tick||strip(currentlogdsn)||tick
      address tso "free  fi(sortin)"
      address tso "free  fi(sortout)"
      address tso "alloc fi(sortin) da("arch") shr unit(tape) " ,
                  "label(sl) position(2) vol("currentvolume")"
      address tso "alloc fi(sortout) da("arch1") shr "
      signal on error
      say 'Copy archive log on temporary disk ' icnt
      address tso "CALL 'SYS1.SORTLPA(ICEMAN)'"
      signal off error
    end
    else do
      arch = tick||strip(currentlogdsn)||tick
      address tso "free  fi(sortin)"
      address tso "free  fi(sortout)"
      address tso "alloc fi(sortin) da("arch") shr unit(tape) " ,
                  "label(sl) position(2) vol("currentvolume")"
      address tso "alloc fi(sortout) da("arch1") mod "
      signal on error
      say 'Copy archive log on temporary disk ' icnt
      address tso "CALL 'SYS1.SORTLPA(ICEMAN)'"
      signal off error
    end
    rbaendx = currentendrba
  end
end
address tso "free fi(sysprint)"
address tso "free fi(sysut1)"
address tso "free fi(sysabend)"
address tso "free fi(syssumry)"
address tso "free fi(sysin)"
if sysdsn(sysi) = 'OK' then do
  address tso "alloc fi(sysin) da("sysi") shr "
end
else
  address tso "alloc fi(sysin) da("sysi") new ",
              "  dsorg(ps) space(1,1) tracks",
              "  lrecl(80) blksize(27920) recfm(f b)"
address tso "delstack"
queue  '  RBASTART('||rbastartx||')'
queue  '  RBAEND('||rbaendx||')'
queue  '  SUMMARY(ONLY)'
```

```
queue  '  DATAONLY(NO)'
queue  '  SYSCOPY(NO)'
address tso "execio 5 diskw sysin"
address tso "execio Ø diskw sysin(finis"
address tso "alloc fi(sysprint) da(*) "
address tso "alloc fi(sysabend) da(*) "
sumry = tick||userid||'.LOGSUM.PRIV'||tick
if sysdsn(sumry) = 'OK' then
  address tso "alloc fi(syssumry) da("sumry") shr "
else
  address tso "alloc fi(syssumry) da("sumry") new ",
              "  dsorg(ps) space(4ØØØ,1ØØØ) tracks release",
              "  lrecl(131) blksize(279Ø3) recfm(f b a)"
signal on error
address tso "CALL '"||db2load||"(DSN1LOGP)'"
signal off error
address tso "free fi(archive)"
address tso "free fi(sortin)"
address tso "free fi(sortout)"
address tso "free fi(syssumry)"
address tso "free  fi(indd1)"
address tso "alloc fi(indd1) da("sumry") shr"
address tso "delstack"
do forever
  address tso "execio 1 diskr indd1"
  if rc = 2 then do
    signal eof_rtn1
  end
  pull inrec
  if index(inrec,'DSN1151I') > Ø then do
    parse var inrec,
      32 currentconnid 4Ø ,
      49 currentcorrid 57 ,
      7Ø currentauthid 77 ,
      85 currentplan 93
    CONNID     = currentconnid
    CORRID     = currentcorrid
    AUTHID     = currentauthid
    PLAN       = currentplan
    address tso "execio 1 diskr indd1"
    if rc = 2 then do
      signal eof_rtn1
    end
    pull inrec
    parse var inrec,
      26 currentdate 32 ,
      38 currenttime 43 ,
      53 currentdisposit 54 ,
      7Ø currentstatus 71
    DATE     = currentdate
    TIME     = currenttime
```

```
DISPOSIT = currentdisposit
STATUS   = currentstatus
address tso "execio 1 diskr indd1"
if rc = 2 then do
  signal eof_rtn1
end
pull inrec
parse var inrec,
  24 currentstartlog 36 ,
  45 currentendlog 57
STARTLOG   = currentstartlog
ENDLOG     = currentendlog
address tso "execio 1 diskr indd1"
if rc = 2 then do
  signal eof_rtn1
end
pull inrec
address tso "execio 1 diskr indd1"
if rc = 2 then do
  signal eof_rtn1
end
pull inrec
address tso "execio 1 diskr indd1"
if rc = 2 then do
  signal eof_rtn1
end
pull inrec
fnd = Ø
do forever
  address tso "execio 1 diskr indd1"
  if rc = 2 then do
    signal eof_rtn1
  end
  pull inrec
  if index(inrec,'DATABASE') > Ø then do
    parse var inrec,
      29 currentdbid 33 ,
      53 currenttsid 57
    if (TDBID = currentdbid & ,
        TTSID = currenttsid) then do
      fnd = 1
    end
  end
  else do
    leave
  end
end   /* do forever for database */
if fnd = 1 then do
  fnd = Ø
  'TBADD 'tbnam1''
end
```

```
      end   /* end if for dsn1151i */
    end   /* do forever for dsn1151i */
    eof_rtn1:
      if rc = 2 then say 'end-of-file'
      address tso "execio Ø diskr indd1(finis"
      address tso "free  fi(indd1)"
      address tso "delete "sumry
    eof_rtn:
      if rc = 2 then say 'end-of-file'
      address tso "execio Ø diskr indd(finis"
      address tso "free  fi(indd)"
      address tso "delete "indsn
      signal endproc_s_det
error:
    say 'error on line:' sigl '  ,rc:' rc
endproc_s_det:
  return

  process_s1:
    'CONTROL DISPLAY SAVE    '
    address tso "free fi(archive)"
    address tso "free fi(sysprint)"
    address tso "free fi(sysabend)"
    address tso "free fi(sysin)"
    if sysdsn(sysi) = 'OK' then do
      address tso "alloc fi(sysin) da("sysi") shr "
    end
    else
      address tso "alloc fi(sysin) da("sysi") new ",
                  "  dsorg(ps) space(1,1) tracks",
                  "  lrecl(8Ø) blksize(2792Ø) recfm(f b)"
    address tso "delstack"
    queue  '  RBASTART('||STARTLOG||')'
    queue  '  RBAEND('||ENDLOG||')'
    queue  '  DBID('||TDBID||')'
    queue  '  OBID('||TTSID||')'
    queue  '  DATAONLY(YES)'
    queue  '  SUBTYPE(1)'
    address tso "execio 6 diskw sysin"
    address tso "execio Ø diskw sysin(finis"
    address tso "alloc fi(archive) da("arch1") shr "
    address tso "alloc fi(sysabend) da(*) "
    detl = tick||userid||'.LOGDET.PRIV'||tick
    if sysdsn(detl) = 'OK' then
      address tso "alloc fi(sysprint) da("detl") shr "
    else
      address tso "alloc fi(sysprint) da("detl") new ",
                  "  dsorg(ps) space(1ØØØ,1ØØ) tracks",
                  "  lrecl(131) blksize(279Ø3) recfm(f b a)"
    signal on error
```

```
address tso "CALL '"||db2load||"(DSN1LOGP)'"
signal off error
address tso "free fi(archive)"
outdsn   = tick||userid||'.LOGOUT.PRIV'||tick
if sysdsn(outdsn) = 'OK' then
  address tso "alloc fi(logout) da("outdsn") shr "
else do
  address tso "alloc fi(logout) da("outdsn") new ",
              "  dsorg(ps) space(4000,1000) tracks release",
              "  recfm(F B) lrecl(132) blksize(27984)"
end
address tso "free  fi(indd1)"
address tso "alloc fi(indd1) da("detl") shr"
address tso "delstack"
call process_header
do forever
  address tso "execio 1 diskr indd1"
  if rc = 2 then do
    signal eof_rtn2
  end
  pull inrec
  if index(inrec,'DSN1212I') > 0 then do
    leave
  end
end
do forever
  address tso "execio 1 diskr indd1"
  if rc = 2 then do
    signal eof_rtn2
  end
  pull inrec
  if index(inrec,'DSN1213I') > 0 then do
    leave
  end
  if index(inrec,'URID(') > 0 then do
    address tso "execio 1 diskr indd1"
    if rc = 2 then do
      signal eof_rtn2
    end
    pull inrec
    address tso "execio 1 diskr indd1"
    if rc = 2 then do
      signal eof_rtn2
    end
    pull inrecsubt
    address tso "execio 1 diskr indd1"
    if rc = 2 then do
      signal eof_rtn2
    end
    pull inrec
```

```
      address tso "execio 1 diskr indd1"
      if rc = 2 then do
        signal eof_rtn2
      end
      pull inrec
      address tso "execio 1 diskr indd1"
      if rc = 2 then do
        signal eof_rtn2
      end
      pull inrec
      address tso "execio 1 diskr indd1"
      if rc = 2 then do
        signal eof_rtn2
      end
      pull inrec
      address tso "execio 1 diskr indd1"
      if rc = 2 then do
        signal eof_rtn2
      end
      pull inrec
      parse var inrec,
        22 currentobid 26
      if TTBID = currentobid then do
        start = index(inrecsubt, 'SUBTYPE') + 8
        len   = index(inrecsubt, ')') - start
        subt  = substr(inrecsubt, start, len)
        select
          when index(subt, 'DELETE') > 0 then call process_id
          when index(subt, 'INSERT') > 0 then call process_id
          when index(subt, 'UPDATE') > 0 then call process_u
          otherwise
        end
      end
    end   /* end if for URID( */
  end   /* do forever */
  eof_rtn2:
    if rc = 2 then say 'end-of-file'
    address tso "execio 0 diskr indd1(finis"
    address tso "free  fi(indd1)"
    queue " *** End-of-detailed-report ***"
    address tso "execio 1 diskw logout"
    address tso "execio 0 diskw logout(finis"
    address tso "free fi(logout)"
    address ispexec 'BROWSE DATASET('outdsn')'
    address tso "delete "outdsn

  address tso "delete "detl
  address tso "delete "sysi
  'CONTROL DISPLAY RESTORE '
return
```

```
process_id:
  parse var inrec,
    44 currentdata 84
  data = space(currentdata, Ø)
  call process_all
  call process_idu 'INSDEL'
return

process_u:
  if index(subt, 'NOT IN-PLACE') > Ø then do
    parse var inrec,
      31 currentoffs 35 ,
      35 currentldatau 39 ,
      4Ø currentldatar 44 ,
      44 currentdata 84
  end
  else do
    parse var inrec,
      31 currentoffs 35 ,
      35 currentdata 84
  end
  offs = currentoffs
  data = space(currentdata, Ø)
  call process_all
  if index(subt, 'NOT IN-PLACE') > Ø then do
    call process_idu1
  end
  else do
    call process_idu 'UPD'
  end
return

process_all:
  drop datapom
  datapomi = Ø
  do forever
    address tso "execio 1 diskr indd1"
    if rc = 2 then do
      leave
    end
    pull inrec
    parse var inrec,
       7 currentaddr 11 ,
      13 currentdata 84
    if index(currentaddr, ' ') > Ø then do
      leave
    end
    datapomi = datapomi + 1
    datapom.datapomi = space(currentdata, Ø)
  end
```

```
  do datapomj = 1 to datapomi
    data = data || datapom.datapomj
  end
  queue ""
  queue subt
  queue ""
  address tso "execio 3 diskw logout"
return

process_header:
  queue "DB2 subsystem: " || DSN8SSID
  queue "       DBNAME: " || DBNAME
  queue "       TSNAME: " || TSNAME
  queue "       TBNAME: " || TCREATOR||"."||TBNAME
  queue ""
  queue "STARTLOG: " || STARTLOG
  queue "  ENDLOG: " || ENDLOG
  queue ""
  queue "CONNID   CORRID   AUTHID   PLAN     DATE   TIME  DISP STATUS"
  queue "======== ======== ======== ======== ====== ===== ==== ======"
  queue left(CONNID,9)||left(CORRID,9)||left(AUTHID,9)||,
        left(PLAN,9)||left(DATE,7)||left(TIME,6)||left(DISPOSIT,5)||,
        left(STATUS,6)
  queue ""
  queue "****************** Begin-of-detailed-report ****************"
  address tso "execio 13 diskw logout"
return

process_idu:
  parse arg op
  uinpl = Ø
  if op = 'UPD' then do
    posi = X2D(offs) - 5
    pos = 1
    j = 1
    do i = 1 to ltab
      if NULLS.i = "Y" then do
        pos = pos + 1
      end
      if COLTYPE.i = "DECIMAL" then do
        p1 = (LENGTH.i + 1) % 2
      end
      else do
        if COLTYPE.i = "VARCHAR" then do
          if LENGTH.i <= 255 then do
            p2 = substr(data, pos, 1)
            p3 = substr(data, pos + 1, 1)
            p1 = (X2D(p2) * 16 + X2D(p3) + 1) * 2
          end
          else do
```

```
            p2 = substr(data, pos, 1)
            p3 = substr(data, pos + 1, 1)
            p4 = substr(data, pos + 2, 1)
            p5 = substr(data, pos + 3, 1)
            p1 = (X2D(p2) * 4Ø96 + X2D(p3) * 256 + ,
                  X2D(p4) * 16 + X2D(p5) + 2) * 2
          end
        end
        else do
          p1 = LENGTH.i
        end
      end
      pos = pos + p1
      if pos > posi then do
        leave
      end
      j = i + 1
    end
  end
  else do
    j = 1
    posi = Ø
    posj = Ø
  end
  k = 1
  if op = 'UPD' then do
    posur = length(data) % 2
  end
  else do
    posur = length(data)
  end
  pos = 1
  data1 = substr(data, 1, posur)
  bound = Ø
  do i = j to ltab
    call process_idu2 data1 posur posi posj "UNDO" uinpl
    if pos > posur then leave
  end
  if op = 'UPD' then do
    pos = 1
    data1 = substr(data, posur + 1, posur)
    bound = Ø
    do i = j to ltab
      call process_idu2 data1 posur posi posj "REDO" uinpl
      if pos > posur then leave
    end
  end
  do i = 1 to bound
    if op = 'UPD' then do
      queue "   " || URNAME.i
```

```
      queue "          REDO: " || undo.i
      queue "          UNDO: " || redo.i
      address tso "execio 3 diskw logout"
    end
    else do
      queue "   "||left(URNAME.i,18)||" : "||undo.i
      address tso "execio 1 diskw logout"
    end
  end
  queue ""
  address tso "execio 1 diskw logout"
return

process_idu1:
  uinpl = 1             /* update not-in place */
  posi = X2D(offs) - 5
  posu = X2D(currentldatau) * 2
  posr = X2D(currentldatar) * 2
  pos = 1
  j = 1
  do i = 1 to ltab
    indnull = 0
    if NULLS.i = "Y" then do
      pos = pos + 1
      indnull = 1
    end
    if COLTYPE.i = "DECIMAL" then do
      p1 = (LENGTH.i + 1) % 2
    end
    else do
      if COLTYPE.i = "VARCHAR" then do
        if pos < length(data) then do
          if LENGTH.i <= 255 then do
            p2 = substr(data, pos, 1)
            p3 = substr(data, pos + 1, 1)
            p1 = (X2D(p2) * 16 + X2D(p3) + 1) * 2
          end
          else do
            p2 = substr(data, pos, 1)
            p3 = substr(data, pos + 1, 1)
            p4 = substr(data, pos + 2, 1)
            p5 = substr(data, pos + 3, 1)
            p1 = (X2D(p2) * 4Ø96 + X2D(p3) * 256 + ,
                  X2D(p4) * 16 + X2D(p5) + 2) * 2
          end
        end
        if p1 > posu / 2 then do
          p1 = Ø
          pos = pos - indnull
        end
```

```
      end
      else do
        p1 = LENGTH.i
      end
    end
    pos = pos + p1
    k = 0
    if pos > posi then do
      leave
    end
    if pos = posi then do
      k = 1
      leave
    end
    j = i + 1
  end
  data1 = substr(data, 1, posu)
  posj = pos
  pos = 1
  posur = posu
  bound = 0
  do i = j to ltab
    call process_idu2 data1 posur posi posj "UNDO" uinpl
    if pos > posur then leave
  end
  data1 = substr(data, posu + 1, posr)
  pos = 1
  posur = posr
  bound = 0
  do i = j to ltab
    call process_idu2 data1 posur posi posj "REDO" uinpl
    if pos > posur then leave
  end
  do i = 1 to bound
    queue "   " || URNAME.i
    queue "         REDO: " || undo.i
    queue "         UNDO: " || redo.i
    address tso "execio 3 diskw logout"
  end
  queue ""
  address tso "execio 1 diskw logout"
return

process_idu2:
  parse arg data1 posur posi posj op2 uinpl
    d1 = ""
    if i > j - k then do
      if NULLS.i = "Y" then do
        if substr(data1, pos, 2) = "FF" then do
          d1 = "NULL"
```

```
    end
    pos = pos + 2
  end
end
select
  when COLTYPE.i = "DECIMAL" then do
    p1 = ((LENGTH.i + 1) % 2) * 2
    q1 = SCALE.i
  end
  when COLTYPE.i = "VARCHAR" then do
    if LENGTH.i <= 255 then do
      p2 = substr(data1, pos, 1)
      p3 = substr(data1, pos + 1, 1)
      p1 = (X2D(p2) * 16 + X2D(p3) + 1) * 2
    end
    else do
      p2 = substr(data1, pos, 1)
      p3 = substr(data1, pos + 1, 1)
      p4 = substr(data1, pos + 2, 1)
      p5 = substr(data1, pos + 3, 1)
      p1 = (X2D(p2) * 4Ø96 + X2D(p3) * 256 + ,
            X2D(p4) * 16 + X2D(p5) + 2) * 2
    end
    q1 = Ø
  end
  otherwise do
    p1 = LENGTH.i * 2
    q1 = Ø
  end
end
if d1 <> "NULL" then do
  if i = j & uinpl = 1 then do
    if (COLTYPE.i = "CHAR") then do
      d1 = substr(data1, pos, p1 - (posi + 1) % 2)
    end
    else do
      if (COLTYPE.i = "VARCHAR") then do
        d1 = substr(data1, pos, posur)
      end
      else do
        d1 = substr(data1, pos, (posj - posi) * 2)
      end
    end
  end
  else do
    d1 = substr(data1, pos, p1)
  end
  select
    when (COLTYPE.i = "INTEGER" | COLTYPE.i = "SMALLINT") then do
      NUMERIC DIGITS 1Ø
```

```
  if i = j & uinpl = 1 & length(d1) < p1 then do
    d1 = "X'" || d1 || "'"
  end
  else do
    a = x2b(d1)
    b = \ substr(a, 1, 1) || substr(a, 2, length(a) - 1)
    c = b2x(b)
    if COLTYPE.i = "INTEGER" then do
      d1 = x2d(c,8)
    end
    else do
      d1 = x2d(c,4)
    end
  end
end
when COLTYPE.i = "DECIMAL" then do
  NUMERIC DIGITS 50
  if i = j & uinpl = 1 & length(d1) < p1 then do
    if (VERIFY(d1,'1234567890') > 0) then do
      a = ""
      do ii = 1 to length(d1)
        a = a || (15 - x2d(substr(d1, ii, 1)))
      end
      d1 = a
    end
    d1 = left(d1, length(d1) - q1) || "," ||,
         right(d1, q1)
  end
  else do
    a = substr(d1, 2, length(d1) - 1)
    if substr(d1, 1, 1) <> "F" then do
      a = -b2x(translate(x2b(a), '01', '10'))
    end
    else do
      a = strip(a,'L',0)
      if a = "" then do
        a = 0
      end
    end
    if q1 = 0 then do
      d1 = a
    end
    else do
      if a > 0 then do
        d1 = left(a, length(a) - q1) || "," ||,
             right(a, q1)
      end
      else do
        a = substr(a, 2, length(a) - 1)
        d1 = left(a, length(a) - q1) || "," ||,
```

```
            right(a, q1)
          d1 = "-" || d1
        end
      end
    end
  end
when COLTYPE.i = "CHAR"  then do
  d1 = "'" || strip(X2C(d1),'T') || "'"
end
when COLTYPE.i = "VARCHAR" then do
  if i = j & uinpl = 1 then do
    d1 = "'" || strip(X2C(d1),'T') || "'"
  end
  else do
    if (d1 = "00" | d1 = "0000") then do
      if d1 = "00" then do
        p1 = p1 + 2
      end
      d1 = ""
    end
    else do
      d1 = "'" || strip(X2C(substr(d1, 5, p1 - 4)),'T') || "'"
    end
  end
end
when COLTYPE.i = "DATE" then do
  d1 = substr(d1, 7, 2) || "." || substr(d1, 5, 2) || "." ||,
       substr(d1, 1, 4)
end
when COLTYPE.i = "TIME" then do
  d1 = substr(d1, 1, 2) || ":" || substr(d1, 3, 2) || ":" ||,
       substr(d1, 5, 2)
end
when COLTYPE.i = "TIMESTMP" then do
  d1 = substr(d1, 1, 4) || "-" || substr(d1, 5, 2) || "-" ||,
       substr(d1, 7, 2) || "-" || substr(d1, 9, 2) || "." ||,
       substr(d1,11, 2) || "." || substr(d1,13, 2) || "." ||,
       substr(d1,15, 6)
end
otherwise do   /* float */
  NUMERIC DIGITS 50
  if i = j & uinpl = 1 then do
    d1 = "X'" || d1 || "'"
  end
  else do
    a = x2b(d1)
    if substr(a, 1, 1) = "1" then do
      a = "0" || substr(a, 2, length(a) - 1)
      s = 0
    end
```

```
            else do
              a = translate(a, '01', '10')
              s = -1
            end
            e = x2d(b2x(substr(a, 2, 7))) - 64
            S1 = b2x(substr(a, 9, length(a) - 8))
            b = 0
            do ii = 1 to length(S1)
              b = b + x2d(substr(S1, ii, 1)) / (16 ** ii)
            end
            d1 = (-1) ** s * b * 16 ** e
            if p1 = 8 then do
              d1 = FORMAT(d1,,7,2,2)
            end
            else do
              d1 = FORMAT(d1,,16,2,2)
            end
          end
        end
      end
    end
    if i = j & uinpl = 1 then do
      if (COLTYPE.i = "CHAR") then do
        pos = pos + p1 - (posi + 1) % 2
      end
      else do
        if (COLTYPE.i = "VARCHAR") then do
          pos = pos + posur
        end
        else do
          pos = pos + (posj - posi) * 2
        end
      end
    end
    else do
      pos = pos + p1
    end
    bound = bound + 1
    if op2 = "UNDO" then do
      URNAME.bound = NAME.i
      undo.bound = d1
    end
    else do
      URNAME.bound = NAME.i
      redo.bound = d1
    end
  return
```

NEON Systems has announced that it has added BEA WebLogic Platform 7.0 support to its ShadowConnect and ShadowDirect products.

With this integration, the NEON and BEA solutions provide JCA, JDBC, or ODBC access to mainframe data sources and transactional environments supporting CICS, IMS, DB2,ADABAS, Natural, IDMS, and flat files.

ShadowDirect J2EE connectors for z/OS are now certified for BEA WebLogic.

For further information contact:
NEON Systems**,** 14100 Southwest Freeway, Suite 500, Sugar Land, TX 77478, USA.
Tel: (281) 491 4200.
NEON Systems UK, No 1 High Street, Windsor, Berkshire SL4 1LD, UK.
Tel: (01753) 752800.
URL: http://www.neonsys.com.

* * *

Following its June acquisition of Metamerge, IBM has announced the LDAP-based Directory Server 4.1, promising a software infrastructure for identifying enterprise resources, controlling access to networked systems, and deploying Web services securely across leading operating systems, including Linux, for Intel and zSeries servers. The metadirectory software aggregates directory information that may be stored in directories or databases provided by multiple software vendors and running on a variety of operating systems.

Also introduced is new Metadirectory and Directory Professional Services to help with the analysis, design, and deployment of enterprise directory and metadirectory applications.

Built on DB2 and supporting AIX, HP-UX, Linux, Solaris, and Windows 2000, the software provides a common identity infrastructure for Web applications that can use Directory Server to identify network users and resources.

It also provides directory functions that link user data so that security applications get simplified access to the data, to help provide user provisioning and access management across Web and legacy applications.
It's out now for free as a download.

For further information contact your local IBM representative.
URL: http://www.ibm.com/software.

* * *

Syncsort has announced Version 2.1.5 of its Backup Express enterprise back-up and restore software, which includes enhanced NDMP features, improved flexibility of media and job control, and advanced remote deployment, device auto-configuration, and cluster support.

There's expanded platform support for database interfaces including DB2, Lotus Notes, and Sybase, plus new NetWare node copy function support.

For further information contact:
Syncsort, 50 Tice Boulevard, Woodcliff Lake, NJ 07677, USA.
Tel: 201 930 8200.
URL: http://www.syncsort.com/bexpress/bexpress29.htm.