92

DB2 update

June 2000

In this issue

3 Capturing accounting information

15 DB2 Version 5 catalog statistics

17 Using LSTCAT output to generate ALTER SQL statements

23 DB2 PLAN_TABLE – access and maintenance

32 Image copy, DSNTIAUL copy, and disaster recovery of DB2 objects – part 2

48 DB2 news

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

Contributions
Articles published in *DB2 Update* are paid for at the rate of £170 ($250) per 1000 words and £90 ($140) per 100 lines of code for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

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

Editor
Trevor Eddolls

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

Capturing accounting information

In order to tune application programs, many DBAs are using a third-party monitoring program. If there is no monitoring tool, SMF data should be used. The process of handling SMF data is complicated.

This ALC program gives accounting information in a real-time environment – like a monitoring tool.

The program starts a trace with an Online Performance (OP) destination, and captures data from OP buffers, then formats the report.

Note:

- The user must have DB2 TRACE authority.
- Because the report is printed in a JES spool, you must be careful if you have a heavily-used DB2 system.
- If other programs are using the OP buffer, you need to control the OP buffers.
- To stop the batch job, you must issue an MVS cancel command.
- To stop the previously invoked trace, you must issue the '-sto trace(a) tno(#)' DB2 command.

Example output is shown in Figure 1.

ASEMBLE AND EXECUTING JCL

```
//JOBLIB  DD  DSN=DB251Ø.SDSNLOAD,DISP=SHR
//*
//PREPUNL EXEC DSNHASM,MEM=DB2CPACT,
//         PARM.PC='HOST(ASM),STDSQL(NO)',
//         PARM.ASM='OBJECT,NODECK',
//         PARM.LKED=(MAP,LET,LIST)
//PC.DBRMLIB   DD DSN=DB2T.DBRMLIB(DB2CPACT),
//             DISP=SHR
//PC.SYSLIB    DD DSN=DB251Ø.SDSNSAMP,
//             DISP=SHR
//PC.SYSIN     DD DSN=DB2T.ASMLIB(DB2CPACT),
//             DISP=SHR
//ASM.SYSLIB   DD
//             DD DSN=DB251Ø.SDSNMACS,
*AUTHID/* CORR-ID  /*CONNID/**PLAN*/**E-TIME**/**C-TIME**/*WAIT I/O*/*WAIT I/O2/
LICJ9HS PTØØIQXC    ASECICTJLNIPQXCØØØ:ØØ:ØØ.21ØØ:ØØ:ØØ.ØØØØ:ØØ:ØØ.2ØØØ:ØØ:ØØ.ØØ
 ** PACKAGE OR DBRM OF ABOVE PLAN **     INCHEONT1      IAQ              LNIP
 ** PACKAGE OR DBRM OF ABOVE PLAN **     INCHEONT1      IAB              LNIS

*COMMIT*/*SELECT/*INSERT/*UPDATE/*DELETE/*FETCH*/*GETPG*/*BP/ *GP/ *SR/ *SP/ *LP
    1       3      Ø      Ø      Ø      Ø      1ØBPØØ
QXCØ      CON-TOCKEN: ]   {   DB2-ETIME:ØØ:ØØ:ØØ.1  DB2-WTIME:ØØ:ØØ:ØØ.1
BMSG      CON-TOCKEN: !       DB2-ETIME:ØØ:ØØ:ØØ.Ø2 DB2-WTIME:ØØ:ØØ:ØØ.Ø2
```

Figure 1: Example output

```
//              DISP=SHR
//LKED.SYSLMOD DD DSN=DB2T.LOADLIB(DB2CPACT),
//              DISP=SHR
//LKED.SYSIN   DD *
  INCLUDE SYSLIB(DSNELI)
  NAME DB2CPACT(R)
//*
```

```
//* BIND AND GRANT EXECUTE AUTHORITY TO someone
//*
//BINDUNL EXEC PGM=IKJEFTØ1,DYNAMNBR=2Ø,COND=(4,LT)
//DBRMLIB  DD  DSN=DB2T.DBRMLIB,
//             DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2T)
  BIND PLAN(DB2CPACT) MEM(DB2CPACT) ACT(REP) ISOLATION(CS)  -
       LIB('DB2T.DBRMLIB')
  RUN PROGRAM(DSNTIAD)  PLAN(DSNTIAD) -
       LIB('DB2T.RUNLIB.LOAD')
  END
//SYSIN    DD  *
   GRANT EXECUTE ON PLAN DB2CPACT TO XXXXXX;
//MONITOR   EXEC PGM=IKJEFTØ1,
//        TIME=144Ø,COND=(4,LT),
//        DYNAMNBR=3Ø,
//        REGION=4M
//STEPLIB  DD  DSN=DB251Ø.SDSNLOAD,DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//REPORT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSTSIN  DD  *
DSN SYSTEM(DB2T)
RUN  PROGRAM(DB2CPACT) PLAN(DB2CPACT)      -
     LIB('DB2T.LOADLIB')
END
/*
//
```

DB2CPACT

```
- PROGRAM SOURCE CODE
         TITLE 'DB2 ACCOUNTING CAPTURE PROGRAM'
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
*  DB2CPACT : REAL-TIME ACCOUNTING INFORMATION CAPTURE               *
*  FUNCTION :                                                        *
*    - START ACCOUNTING TRACE IN PROGRAM                             *
*    - CAPTURE ACCOUNTING INFORMATION FROM 'ONLINE PERFORMANCE BUFFER'*
*    - FORMATTING REPORT                                             *
*  ** NOTE)                                                          *
*    - YOU MUST CHANGE  'OWNR'  FIELD WITH YOUR LOGON ID             *
*      AND HAVE '-STA TRACE' DB2 AUTHORITY.                          *
*    - ONLINE PERFORMANCE BUFFER (OP1) IS USED BY THIS PROGRAM.      *
*      IF YOU WILL USE A SPECIFIC OP#, CHANGE IT.                    *
*    - THIS IS A LOOPING PROGRAM. TO STOP, YOU MUST CANCEL THIS JOB.  *
*      AND YOU MUST ISSUE THE '-STO TRACE' COMMAND TO STOP THE TRACE. *
*    - SOME OUTPUT FIELDS ARE IN HEX FORMAT.                         *
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + *
*  PSEUDOCODE                                                        *
*    OPEN FILES                                                      *
*    WRITE REPORT HEADER OUTPUT                                      *
*    START TRACE                                                     *
*      . USE GETMAIN TO OBTAIN A STORAGE SAME WITH BUFSIZE           *
*      . START TRACE WITH DEST=OPX                                   *
*      . INDICATE TO WAKE UP THIS ROUTINE BY A POST                  *
*        WHENEVER THE BUFFER IS 2Ø% FULL                             *
*      . WAIT FOR THE BUFFER TO BE POSTED                            *
*    READ TRACED DATA FROM BUFFER                                    *
*      . CALL IFI TO OBTAIN THE BUFFER DATA VIA A READA REQUEST      *
*      . CHECK THE STATUS IF THE READA WAS SUCCESSFUL                *
*    FORMAT THE OUTPUT DATA                                          *
*    LOOP BACK TO THE WAIT                                           *
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + *
*  REGISTERS                                                         *
*     R2          IFI RETURN AREA                                    *
*     R3   IFCA   INSTRUMENTATION FACILITY COMMUNICATION AREA        *
*     R4   QWAØ   SELF DEFINE SECTION                                *
*     R5   QWHS   PRODUCTION SECTION - STANDARD HEADER               *
*          QWHC   PRODUCTION SECTION - CORRELATION HEADER            *
*          QWAC   DATA SECTION                                       *
*          QXST   SQL  SECTION                                       *
*          QBAC   BUFFER MANAGER SECTION                             *
*          QPAC   BUFFER MANAGER SECTION                             *
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + *
*  DD CARDS     :                                                    *
*          SYSPRINT -    MESSAGE DD                                  *
*          REPORT   -    OUTPUT DD                                   *
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + *
*  MACROS       :                                                    *
*          DSNDIFCA      IFCA MAPPING MACRO                          *
*          DSNDWBUF      IFC BUFFER INFORMATION BLOCK                *
*          DSNDWQAL      IFC QUALIFICATION BLOCK                     *
*          DSNDQWAS      ACCOUNTING MAPPING MACRO                    *
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + *
*  OUTPUT FIELD :                                                    *
*   1) PLAN INFORMATION :                                            *
*          QWHCAID  -    AUTHORIZATION ID                            *
*          QWHCCV   -    CORRELATION ID                              *
*          QWHCCN   -    CONNECTION NAME                             *
*          QWHCPLAN -    PLAN NAME                                   *
*          QWACASC  -    ELAPSED TIME IN DB2                         *
*          QWACAJST -    TCB TIME IN DB2                             *
*          QWACAWTI -    I/O WAIT TIME                               *
*          QWACAWTR -    ASYNCH READ WAIT TIME                       *
*          QWHSLUCC -    COMMIT COUNT                 (HEX OUTPUT)   *
*          QXSELECT -    # OF SELECT SQL                             *
*          QXINSRT  -    # OF INSERT SQL                             *
*          QXUPDTE  -    # OF UPDATE SQL                                    *
*          QXDELET  -    # OF DELETE SQL                                    *
*          QXFETCH  -    # OF FETCH  SQL                                    *
*          TOTGET   -    TOTAL GET PAGE COUNT                               *
*   2) BUFFER INFORMATION                                                   *
*          QBACPID  -    BUFFER POOL ID                                     *
*          QBACGET  -    # OF GET PAGE                 (HEX OUTPUT)    *
*          QBACRIO  -    # OF SYNCRONOUS I/O           (HEX OUTPUT)    *
*          QBACSEQ  -    # OF SEQUENTIAL PREFETCH I/O (HEX OUTPUT)    *
*          QBACLPF  -    # OF LIST PREFETCH I/O        (HEX OUTPUT)    *
*          QBACDPF  -    # OF DYNAMIC PREFETCH I/O     (HEX OUTPUT)    *
*   3) PACKAGE INFORMATION                                                  *
*          QPACRECN  -   # OF PACKAGE OR DBRM                               *
*          QPACLOCN  -   LOCATION NAME                                      *
*          QPACCOLN  -   COLLECTION ID                                      *
*          QPACPKID  -   PACKAGE NAME                                       *
*          QPACCONT  -   CONSISTENCY TOCKEN                                 *
*          QPACSCT   -   PACKAGE ELAPSE TIME                                *
*          QPACAWTI  -   PACKAGE WAIT TIME                                  *
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + *
*  REFERENCE    :                                                           *
*     'ADMINISTRATION GUIDE'   APPENDIX E. PROGRAMMING FOR THE IFI    *
*                                                                           *
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
ØØ
*--- ENTRY AND SETUP                   ---*
         SPACE
DB2CPACT CSECT
         SAVE  (14,12)
         LR    R12,R15
         USING DB2CPACT,R12
         ST    R13,SAVE+4
         LA    R8,SAVE
         ST    R8,8(R13)
         LR    R13,R8
         SPACE
*---  OPEN DD                          ---*
         OPEN  (OUTUT1,(OUTPUT),REPORT,(OUTPUT))
*---  WRITE REPORT HEADER              ---*
         BAL   R14,HEADRTN
*---  START TRACE                      ---*
         BAL   R14,STATRAC
*---  READ TRACE AND FORMAT REPORT     ---*
         BAL   R14,READRTN
*---  IF YOU DON'T WANT TO LOOP PGM, THE FOLLOWINGS ARE USED. ---*
CLOSRTN  DS    ØH
         CLOSE (OUTUT1,,REPORT)
         L     R13,SAVE+4
         L     R15,=F'8'
         RETURN (14,12),RC=(15)
*---                                   ---*
```

```
*---  ISSUE TRACE COMMAND                ---*
*---                                      ---*
STATRAC  DS    ØH
         ST    R14,STATSAVE
*---                                      ---*
*---  IFCA AREA INITIALIZATION           ---*
*---                                      ---*
         LA    R2,IFCAAREA
         USING IFCA,R2
         MVC   IFCALEN(2),LENIFCA         MOVE IFCA LENTH
         MVC   IFCAID(4),IFCAEYE          MOVE CHARACTER 'IFCA'
         MVC   IFCAOWNR(4),OWNR           MOVE 'OWNR'
         LA    R2,BUFAREA
         USING WBUF,R2
         MVC   WBUFLEN(2),LENWBUF         MOVE WBUF LENTH
         MVC   WBUFEYE(4),WBUFEYE1        MOVE CHARACTER 'WBUF'
         MVC   WBUFECB(4),ECB1ADDR        EDB ADDRESS
         MVC   WBUFBC(4),BUFCT
         DROP  R2
         GETMAIN EC,LV=8192,A=STOADDR,LOC=BELOW
         L     R2,STOADDR
         A     R2,=F'8'
         ST    R2,RETADDR
         MVC   Ø(4,R2),=F'4Ø88'
*---  ISSUE START TRACE                  ---*
         CALL DSNWLI,(COMMAND,IFCAAREA,(R2),OUTAREA,BUFAREA),VL
*---                                      ---*
*---  CHECK RETURN AND RESON CODE        ---*
*---                                      ---*
         LA    R3,IFCAAREA
         USING IFCA,R3
         CLC   IFCARC1,=F'Ø'              CHECK RETURN CODE
         BE    PONGØ
         MVC   OUTDATA(4),IFCARC1
PONGØ    CLC   IFCARC2,=F'Ø'
         BE    STATRXT
         MVC   OUTDATA+4(4),IFCARC2
         MVC   OUTDATA+1Ø(24),=C': CHECK LEFT REASON CODE'
         PUT   OUTUT1,OUTDATA
         B     CLOSRTN
STATRXT  DS    ØH
         DROP  R3
         MVC   OUTDATA(255),RETADDR
         PUT   OUTUT1,OUTDATA
         L     R14,STATSAVE
         BR    R14
         SPACE
*---                                      ---*
READRTN  DS    ØH
         WAIT  ECB=ECB1
         MVC   ECB1(4),=F'Ø'              CLEAR POST FLAG
         L     R2,RETADDR
         MVC   Ø(4,R2),=F'8184'         BUFFER SIZE
*---                                    ---*
         CALL  DSNWLI,(READA,IFCAAREA,(R2)),VL
*---                                    ---*
*---  CHECK RETURN AND RESON CODE       ---*
         LA    R3,IFCAAREA
         USING IFCA,R3
         CLC   IFCARC1,=F'Ø'
         BE    PONG1
         MVC   OUTDATA(4),IFCARC1
PONG1    CLC   IFCARC2,=F'Ø'
         BE    READRXT
         MVC   OUTDATA+4(4),IFCARC2
         MVC   OUTDATA+8(2ØØ),IFCABM
         PUT   OUTUT1,OUTDATA
         B     CLOSRTN
READRXT  DS    ØH
         MVC   Ø(2,R2),IFCABM+2
         DROP  R3
         MVC   2(2,R2),=H'Ø'
         BAL   R14,REPTRTN
         B     READRTN
         SPACE
*---  FORMATTING REPORT                 ---*
REPTRTN  DS    ØH
         ST    R14,REPTSAVE
         LR    R4,R2
         LA    R4,8(R4)                 SKIP LENGTH BYTE 4
         USING QWAØ,R4
         LR    R5,R4
         ST    R5,OFFSET                SAVE OFFSET POINT
         A     R5,QWAØ1PSO              STANDARD HEADER OFFSET
         S     R5,=F'4'
         USING QWHS,R5
         CLI   QWHSTYP,QWHSHSØ1         IS TYPE STANDARD ?
         BNE   REPTRXT
*
         LH    R11,QWHSLUCC             COMMIT COUNT
         BAL   R14,CVDRTN               CONVERSION TO ZONE DECIMAL
         MVC   OWHSLUCC,PACK1+12
*
         AH    R5,QWHSLEN
         USING QWHC,R5
         CLI   QWHCTYP,X'2'             IS IT CORRELATION SECTION?
         BNE   REPTRXT
         CLC   QWHCAID,=C'SYSOPR  '     IF SYSTEM PLAN?
         BE    REPTRXT                     SKIP.
         MVC   OWHCAID,QWHCAID          AUTH ID
         MVC   OWHCCV,QWHCCV            CORRELATION ID
         MVC   OWHCCN,QWHCCN            CONNECTION NAME
         MVC   OWHCPLAN,QWHCPLAN        PLAN NAME
         DROP  R5
         CLC   OWHCPLAN,=C'DB2CPACT'    IF MY PROGRAM?
         BE    REPTRXT                     SKIP.
         CLC   OWHCPLAN,=C'        '    IF PLANNAME US NULL?
         BNE   PING                        NO, GO AHEAD
         CLC   QWAØ1R5O,=X'ØØØØØØØØ'    IS IT DDF PROGRAM?
         BE    REPTRXT                     IF DDF SECTION IS NULL
*                                             GO END
PING     L     R5,OFFSET
         A     R5,QWAØ1R1O              CORRELATION OFFSET
         S     R5,=F'4'
         USING QWAC,R5                  CORRELATION SECTION
*
         MVC   TODFROM,QWACASC          MOVE DB2 E-TIME
         BAL   R14,CONVRTN
         MVC   OWACASC,MILITIME         BINARY DB2 ELAPSED TIME SSS.SS
*
         MVC   TODFROM,QWACAJST         MOVE TCB TIME
         BAL   R14,CONVRTN
         MVC   OWACAJST,MILITIME        BINARY DB2 CPU TIME SSS.SS
*
         MVC   TODFROM,QWACAWTI         MOVE WAIT TIME
         BAL   R14,CONVRTN
         MVC   OWACAWTI,MILITIME        BINARY DB2 I/O WAIT TIME
*
         MVC   TODFROM,QWACAWTR         MOVE ASYNCH READ WAIT TIME
         BAL   R14,CONVRTN
         MVC   OWACAWTR,MILITIME        ASYNCH READ I/O WAIT TIME
         DROP  R5
         CLC   QWAØ1R2O,=X'ØØØØØØØØ'    IS IT NULL FUNCTION?
         BNE   NEXTØ                    IF YES, SKIP
         MVC   OXSELECT,=C'       Ø'    INITIALIZE
         MVC   OXINSRT,=C'       Ø'
         MVC   OXUPDTE,=C'       Ø'
         MVC   OXDELET,=C'       Ø'
         MVC   OXFETCH,=C'       Ø'
         MVC   OXGETPG,=C'       Ø'
         B     NEXT1                    IF YES, SKIP
NEXTØ    DS    ØH
         L     R5,OFFSET                SET INITIAL POINT
         A     R5,QWAØ1R2O              OFFSET OF SQL SECTION
         S     R5,=F'4'
         USING QXST,R5
*                                       # OF SELECT STATEMENT
         L     R11,QXSELECT
         BAL   R14,CVDRTN
         MVC   OXSELECT,PACK1+8
*                                       # OF INSERT STATEMENT
         L     R11,QXINSRT
         BAL   R14,CVDRTN
         MVC   OXINSRT,PACK1+8
*                                      # OF UPDATE STATEMENT
         L     R11,QXUPDTE
         BAL   R14,CVDRTN
         MVC   OXUPDTE,PACK1+8
*                                      # OF DELETE STATEMENT
         L     R11,QXDELET
         BAL   R14,CVDRTN
         MVC   OXDELET,PACK1+8
*                                      # OF FETCH  STATEMENT
         L     R11,QXFETCH
         BAL   R14,CVDRTN
         MVC   OXFETCH,PACK1+8
         DROP  R5
*
* BUFFER MANAGER (DSNDQBAC) SECTION
NEXT1    DS    ØH
         CLC   QWAØ1R3O,=X'ØØØØØØØØ'   IS IT NULL FUNCTION?
         BE    NEXT2                   IF YES, SKIP
         L     R5,OFFSET               SET INITIAL POINT
         A     R5,QWAØ1R3O             OFFSET OF BUFFER MANAGER SECTION
         S     R5,=F'4'
         LH    R6,QWAØ1R3N        # OF BUFFER MANAGER DATA SECTION
         LA    R8,REPORTO+137          POINT OF OUTAREA
         SR    R1Ø,R1Ø
LOOPBUFF DS    ØH
         USING QBAC,R5                 BUFFER MANAGER DSECT
         L     R7,QBACPID              BUFFER POOL ID
         CVD   R7,PACKWORK
         UNPK  BPNUM+2(2),PACKWORK+6(2)
         OI    BPNUM+3,X'FØ'           CONVERT TO ZONE DECIMAL
         MVC   Ø(4,R8),BPNUM
*
         A     R1Ø,QBACGET             ADD TO TOTAL GET PAGE
         MVC   5(4,R8),QBACGET         # OF GET PAGE
         MVC   1Ø(4,R8),QBACRIO        # OF SYNC READ I/O
         MVC   15(4,R8),QBACSEQ        # OF SEQ PREFETCH I/O
         MVC   2Ø(4,R8),QBACLPF        # OF LIST PREFETCH I/O
         MVC   25(4,R8),QBACDPF        # OF DYNAMIC PREFETCH I/O
*
         LA    R8,3Ø(R8)               POINTER OF NEXT PRINT
         LH    R9,QWAØ1R3L             LOAD LENGTH OF BUFFER SECTION
         AR    R5,R9                   POINTER OF NEXT BUFFER SECTION
         BCT   R6,LOOPBUFF
NEXT2    DS    ØH
         ST    R1Ø,TOTGET              SAVE TOTAL GET PAGE
*
         L     R11,TOTGET
         BAL   R14,CVDRTN              CONVERSION TO ZONE DECIMAL
         MVC   OXGETPG,PACK1+8
*
```

```
         DROP  R5
         PUT   REPORT,REPORTO          WRITE OUTPUT
         XC    REPORTO,REPORTO         CLEARE
*
         CLC   QWAØ1R8O,=X'ØØØØØØØØ'   IS IT NULL FUNCTION?
         BE    REPTRXT
         L     R5,OFFSET               SET INITIAL POINT
         A     R5,QWAØ1R8O             OFFSET OF BUFFER MANAGER SECTION
         S     R5,=F'4'
         LH    R6,QWAØ1R8N        # OF BUFFER MANAGER DATA SECTION
         LA    R8,REPORTO+4Ø           POINT OF OUTAREA
         SR    R1Ø,R1Ø                 CLEAR R5
LOOPPACK DS    ØH
         USING QPAC,R5
         MVC   REPORTO+1(35),=C'** PACKAGE OR DBRM OF ABOVE PLAN **'
         MVC   Ø(2,R8),QPACRECN        # OF PACKAGE
         MVC   2(16,R8),QPACLOCN       LOCATION ID
         MVC   18(18,R8),QPACCOLN      COLLECTION ID
         MVC   36(18,R8),QPACPKID      PACKAGE NAME
         MVC   54(11,R8),=C'CON-TOCKEN:'
         MVC   65(8,R8),QPACCONT       CONSISTENCY TOKEN
*
         MVC   73(1Ø,R8),=C'DB2-ETIME:'
         MVC   TODFROM,QPACSCT         MOVE PACKAGE E-TIME
         BAL   R14,CONVRTN
         MVC   83(11,R8),MILITIME     BINARY DB2 ELAPSED TIME SSS.SS
*
         MVC   95(1Ø,R8),=C'DB2-WTIME:'
         MVC   TODFROM,QPACAWTI        MOVE PACKAGE W-TIME
         BAL   R14,CONVRTN
         MVC   1Ø5(11,R8),MILITIME     BINARY DB2 ELAPSED TIME SSS.SS
*
         LH    R9,QWAØ1R8L             LOAD LENGTH OF BUFFER SECTION
         AR    R5,R9                   POINTER OF NEXT BUFFER SECTION
         PUT   REPORT,REPORTO          WRITE OUTPUT
         XC    REPORTO,REPORTO         CLEAR
         BCT   R6,LOOPPACK             LOOPING
REPTRXT  DS    ØH
         DROP  R4
         L     R14,REPTSAVE
         BR    R14
         SPACE
*---  CONVERT TIME FORMAT TO DISPLAY   ---*
CONVRTN  DS    ØH
         ST    R14,CONVSAVE
         STCKCONV STCKVAL=TODFROM,CONVVAL=TODTO,                       X
               TIMETYPE=BIN,DATETYPE=YYYYDDD
         UNPK  TODTIME(9),TODTO(5)
         MVC   MILITIME(2),TODTIME
         MVC   MILITIME+3(2),TODTIME+2
         MVC   MILITIME+6(2),TODTIME+4
         MVC   MILITIME+9(2),TODTIME+6
CONVRXT  DS    ØH
         L     R14,CONVSAVE
         BR    R14
         SPACE
*---  CONVERT HEX TO ZONE DECIMAL      ---*
CVDRTN   DS    ØH
         ST    R14,CVDSAVE
         CVD   R11,PACKWORK
         MVC   PACK1,EDIT2
         ED    PACK1,PACKWORK
CVDRXT   DS    ØH
         L     R14,CVDSAVE
         BR    R14
         SPACE
*---  HEADER ROUTINE                   ---*
HEADRTN  DS    ØH
         ST    R14,HEADSAVE
         MVC   REPORTO+Ø(8),=C'*AUTHID/'           AUTHORIZATION ID
         MVC   REPORTO+8(12),=C'* CORR-ID  /'      CORRELATION ID
         MVC   REPORTO+2Ø(8),=C'*CONNID/'          CONNECTION ID
         MVC   REPORTO+28(8),=C'**PLAN*/'          PLAN NAME
         MVC   REPORTO+36(11),=C'**E-TIME**/'      ELAPSE TIME
         MVC   REPORTO+47(11),=C'**C-TIME**/'      CPU TIME
         MVC   REPORTO+58(11),=C'*WAIT I/O*/'      WAIT I/O TIME
         MVC   REPORTO+69(11),=C'*WAIT I/O2/'      WAIT WRITE TIME
         MVC   REPORTO+8Ø(9),=C'*COMMIT*/'         COMMIT COUNT
         MVC   REPORTO+89(8),=C'*SELECT/'          SELECT COUNT
         MVC   REPORTO+97(8),=C'*INSERT/'          INSERT COUNT
         MVC   REPORTO+1Ø5(8),=C'*UPDATE/'         UPDATE COUNT
         MVC   REPORTO+113(8),=C'*DELETE/'         DELETE COUNT
         MVC   REPORTO+121(8),=C'*FETCH*/'         FETCH   COUNT
         MVC   REPORTO+129(8),=C'*GETPG*/'         TOTAL GETPAGE COUNT
         MVC   REPORTO+137(4),=C'*BP/'             BUFFER POOL NAME
         MVC   REPORTO+142(4),=C'*GP/'             GETPAGE / BP
         MVC   REPORTO+147(4),=C'*SR/'             SYNC READ I/O COUNT
         MVC   REPORTO+152(4),=C'*SP/'             SEQUENTIAL PREFETCH
         MVC   REPORTO+157(4),=C'*LP/'             LIST PREFETCH
         MVC   REPORTO+162(4),=C'*DP/'             DYNAMIC PREFETCH
         MVC   REPORTO+168(27),=C'(*: START AND /: END POINT)'
         PUT   REPORT,REPORTO
         XC    REPORTO,REPORTO
HEADRXT  DS    ØH
         L     R14,HEADSAVE
         BR    R14
         SPACE
*---                                   ---*
COMMAND  DC    CL8'COMMAND '
READA    DC    CL8'READA   '
*--- STORAGE OF LENGTH(IFCA) AND PROPERLY INITIALIZED
LENIFCA  DC    AL2(AFTIFCA-IFCA)
```

```
IFCAEYE  DC    C'IFCA'
OWNR     DC    C'XDBJ'
IFCAAREA DC    XL(AFTIFCA-IFCA)'Ø'
*--- STORAGE OF LENGTH(WBUF) AND PROPERLY INITIALIZED
LENWBUF  DC    AL2(AFTWBUF-WBUF)
WBUFEYE1 DC    C'WBUF'
BUFAREA  DC    XL(AFTWBUF-WBUF)'Ø'
*--- STORAGE FOR LENGTH AND RETURNED INFO
RETADDR  DS    A
STOADDR  DS    A
*--- STORAGE FOR LENGTH AND DB2 COMMAND
OUTAREA  DS    ØCL44
OUTLEN   DC    X'ØØ2CØØØØ'        MUST CHANGE WHEN COMMAND IS CHANGED
OUTCMD   DC    CL4Ø'-STA TRACE(A) DEST(OPX) CLASS(1,2,3,7,8)'
SAVE     DS    18F
STATSAVE DC    F'Ø'
REPTSAVE DC    F'Ø'
CONVSAVE DC    F'Ø'
CVDSAVE  DC    F'Ø'
HEADSAVE DC    F'Ø'
OFFSET   DC    F'Ø'
BUFCT    DC    F'2Ø'
ECB1     DC    F'Ø'
ECB1ADDR DC    A(ECB1)
TODFROM  DS    CL8                 WORK AREA FOR TIME CONVERT
TODTO    DS    CL16
TODTIME  DS    CL9
TOTGET   DC    F'Ø'                # OF TOTAL GET PAGE
PACKWORK DS    PL8                 TIME PACK DECIMAL
PACK1    DS    XL16                BUFFER PACK DECIMAL
MILITIME DC    CL11'HH:MM:SS.XX'   TCB & ELAPSE TIME DISPLAY
BPNUM    DC    CL4'BP##'           BUFFER POOL NAME
EDIT2    DC    X'4Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø212Ø'
OUTDATA  DS    CL256
*
REPORTO  DS    ØCL25Ø
OWHCAID  DS    CL8
OWHCCV   DS    CL12
OWHCCN   DS    CL8
OWHCPLAN DS    CL8
OWACASC  DS    CL11
OWACAJST DS    CL11
OWACAWTI DS    CL11
OWACAWTR DS    CL11
OWHSLUCC DS    F                       # COMMIT POINT
OXSELECT DS    D
OXINSRT  DS    D
OXUPDTE  DS    D
OXDELET  DS    D
OXFETCH  DS    D
OXGETPG  DS    D
```

```
         DS    CL12Ø
EREPORTO EQU   (*-REPORTO)
OUTUT1   DCB   DSORG=PS,MACRF=(PM),DDNAME=SYSPRINT,                    *
               RECFM=F,LRECL=256,BLKSIZE=256
IFCADCB  DCB   DSORG=PS,MACRF=(PM),DDNAME=IFCAOUT,                     *
               RECFM=VB,LRECL=8192,BLKSIZE=8196
REPORT   DCB   DSORG=PS,MACRF=(PM),DDNAME=REPORT,                      *
               RECFM=FB,LRECL=25Ø,BLKSIZE=5ØØØ
*--- MACROS
         YREGS
         DSNDIFCA DSNDIFCA_LIST=Y      IFCA MAPPING MACRO
AFTIFCA  EQU   *
         DSNDWBUF                      IFC BUFFER INFORMATION BLOCK
AFTWBUF  EQU   *
         DSNDWQAL                      IFC QUALIFICATION BLOCK
AFTWQAL  EQU   *
         DSNDQWAS DSECT=YES,SUBTYPE=ALL
         END
```

Kim Whang Gi
System Programmer
LG-EDS Systems (South Korea)

DB2 Version 5 catalog statistics

The REXX EXEC CSUPD, published in the article entitled *DB2 catalog statistics update REXX EXEC,* in Issue 78, April 1999 requires changes for Version 5 of DB2.

In Version 5, floating decimal fields were added to cater for large tables.

RUNSTATS will still update the original integer fields and the new floating point fields for non-large tables, and will update the floating decimal fields but not the corresponding integer fields for large tables. The DB2 optimizer now uses the floating point fields.

The changes here do not cater for large tables but are only intended to ensure that statistics entered are copied to the fields used by the optimizer.

The changes are straightforward and are marked in italics.

Alter comments:

```
 /* REXX  */
 /*                                                            */
 /*  This EXEC will retrieve and update catalog statistics     */
 /*  for a given DB2 table.                                    */
 /*                                                            */
 /*  The EXERRC has been altered for Version 5 of DB2 to update */
 /*  the '%CARDF' catalog statistics in line with the '%CARD' */
 /*  statistics. CARDF, therefore, can never be greater than  */
 /*  CARD.                                                     */
 /*                                                            */
```

Find routine:

```
DB_UPDATE:
  signal on failure
 /*-----------------------------------------*/
 /* Update Table data - if data has changed */
 /*-----------------------------------------*/
```

Alter update statement:

```
     UPDT = "UPDATE SYSIBM.SYSTABLES",
            "SET CARD="ROWS",NPAGES="NPAGES",",
            "PCTPAGES="PCTPAGES",CARDF="ROWS
     WHRCLS = "WHERE CREATOR='"CRTR"' AND NAME='"TBNAM"'"
```

Alter update statement:

```
     UPDT = "UPDATE SYSIBM.SYSINDEXES ",
            "SET FIRSTKEYCARD="NFSKCRD",FULLKEYCARD="FLKCRD",",
            "FIRSTKEYCARDF="NFSKCRD",FULLKEYCARDF="FLKCRD",",
            "NLEAF="NLEAF",NLEVELS="NLVLS",CLUSTERRATIO="NCRIO
     WHRCLS = "WHERE CREATOR='"ICRTR"' AND NAME='"INAME"'"
```

Find routine:

```
 /*--------------------*/
 /* Update Column data */
 /*--------------------*/
EA_UPDATE:
```

Alter update statement:

```
        UPDT = "UPDATE SYSIBM.SYSCOLUMNS ",
            "SET COLCARD="NCCARD",LOW2KEY='"NL2KEY"',",
            "HIGH2KEY='"NH2KEY"',COLCARDF="NCCARD
        WHRCLS = "WHERE TBCREATOR='"CRTR"' AND TBNAME='"TBNAM"'",
                 "AND NAME='"UCNAME"'"
```

Liz Page
Independent Consultant (UK) © Xephon 2000

Using LSTCAT output to generate ALTER SQL statements

The GENALTR utility works on the output of the LSTCAT utility. It checks the old PRIQTY and the new PRIQTY and, if they are different, it will generate ALTER SQL statements for that particular object, which could be an indexspace or a tablespace.

Apart from the input dataset, other input required includes:

- A creator name for the index.
- A default percentage for the secondary quantity. (This specifies the percentage of the new primary quantity to be used as the new secondary quantity.)

The input file to the GENALTR utility is coded as an argument and hence it can be conveniently executed from an ISPF 3.4 list panel containing the output of the LSTCAT execution.

There are two output files from the GENALTR utility:

- An ALTER dataset containing ALTER DDL – see ALTOUT sample output below.
- An FRSPC dataset containing free space information – see ALTFPC sample output below.

The output dataset's name has the format PREFIX.USERID.ALTER.*, or it can be a user-specified name.

The summary dataset's name has the format PREFIX.USERID. FRSPC.*.

The key to generating the ALTER statements is the difference between the NPQTY and the PQTY in the LSTCAT output. If the difference is positive, then the ALTER statement is generated.

The NSQTY value is derived as follows: if the input dataset contains a numeric value in the NSQTY field, then that value is used; if it contains the word DEFAULT, then the default secondary percentage

```
 DBNAME   OBJECT  PART VOLSER  NUPGS  PQTY   SQTY  EXTS  SPCALC  SPCUSE   %USE   NPQTY  NSQTY   N%use PART   OBNAME
------------------------------------------------------------------------------------------------------------------------
XXTESTDB TBADETTS  ØØ1 VOLØØ1  1836Ø  288Ø   144Ø    5Ø   7344Ø   7344Ø 1ØØ.ØØ   7488Ø   288Ø   98.Ø7  ØØ1  TBADETTS TN
XXTESTDB TBADBMTS  ØØ1 VOLØØ1   1Ø8Ø  288Ø    72Ø     3    432Ø    432Ø 1ØØ.ØØ    5Ø4Ø   144Ø   85.71  ØØ1  TBASUMTS TN
XXTESTDB TBANOTTS  ØØ1 VOLØØ1   1Ø8Ø  288Ø    72Ø     3    432Ø    432Ø 1ØØ.ØØ    5Ø4Ø   144Ø   85.71  ØØ1  TBANOTTS TN
XXTESTDB TBAISSTS  ØØ1 VOLØØ1   1Ø8Ø  288Ø    72Ø     3    432Ø    432Ø 1ØØ.ØØ    5Ø4Ø   144Ø   85.71  ØØ1  TBAISSTS TN
XXTESTDB TBABAUTS  ØØ1 VOLØØ1   216Ø  288Ø    72Ø     9    864Ø    864Ø 1ØØ.ØØ   1152Ø   144Ø   75.ØØ  ØØ1  TBABAUTS TN
XXTESTDB TBABCLTS  ØØ1 VOLØØ1    36Ø  288Ø    72Ø     1    288Ø    144Ø  5Ø.ØØ    216Ø   144Ø   66.66  ØØ1  TBABCLTS TN
_
```

Figure 1: Example input

specified will be applied to the NPQTY and rounded off to the next higher cylinder boundary.

The utility also prompts the user to perform FREE SPACE analysis. This is done by calling the CHKVTOC utility, which invokes the IBM supplied IEHLIST utility with those volume names and retrieves the necessary information. It then processes the key information to get the free space availability on the volume and reports it back.

Note: the information returned about the free space availabilty on the volume must be analysed carefully. If the additional space being requested on a volume is 150 cylinders and if the utility indicates that the free space available is 150 cylinders or even 175 cylinders, it does not indicate a perfect fit. The additional space requirement is calculated using the new primary quantity being requested. However, when DB2 reorganizes the object, it is preferabe to have the new primary quantity space available in one extent (or a maximum of five extents). If DB2 cannot get this, there could be serious problems, possibly with dataset loss.

Another use for this utility is in re-sizing a test database to hold production volumes of data. The LSTCAT utility can be run on the production databases with a one percent default increase, and then the GENALTR utility can be run on the LSTCAT ouput to generate the ALTER statements for all objects to reflect the correct quantity used. This way we can optimize the space requirements on the TEST database.

Other utilities to generate REORG, image copy, and RUNSTATS JCL using the output from the LSTCAT utility can be written along these lines, thereby aiding productivity.

Example input is shown in Figure 1.

ALTOUT

Sample output:

```
 ALTER TABLESPACE XXTESTDB.TBADETTS
   PRIQTY 7488Ø SECQTY 288Ø ;
 ALTER TABLESPACE XXTESTDB.TBADBMTS
   PRIQTY 5Ø4Ø SECQTY 144Ø ;
 ALTER TABLESPACE XXTESTDB.TBANOTTS
   PRIQTY 5Ø4Ø SECQTY 144Ø ;
 ALTER TABLESPACE XXTESTDB.TBAISSTS
   PRIQTY 5Ø4Ø SECQTY 144Ø ;
 ALTER TABLESPACE XXTESTDB.TBABAUTS
   PRIQTY 1152Ø SECQTY 144Ø ;
 ALTER TABLESPACE XXTESTDB.TBABCLTS
   PRIQTY 216Ø SECQTY 144Ø ;
```

ALTFSPC

Sample output:

```
-------------------------
Volume  Cylreq    Cylfree
-------------------------
VOLØØ1    121     444
-------------------------
```

CHKVTOC

```
/* REXX  */
/*                                                                */
/* Invocation  tso CHKVTOC VOLUMENAME                             */
/*                                                                */
/*  The sysprint dataset is also present and the code can be      */
/*  turned on or off to browse the same                           */

TRACE o

PREFX = SYSVAR(SYSPREF)
PARSE UPPER ARG P_volnam
if strip(P_volnam)='' | strip(P_volnam) = '' then
do
   say 'Proper execution is LSTVTOC VOLUME NAME  ...'
   exit(8)
end

x = outtrap("zap.","*")
CALL P1ØØØ_Allocate_Sysin
CALL P2ØØØ_Allocate_Output
Call P3ØØØ_Execute_IEHLIST
Call P4ØØØ_Clean_up
```

```
x = outtrap("OFF")

EXIT
/*   */
/*   */
P1ØØØ_Allocate_Sysin:
   P_sysin = PREFX||'.'||USERID()||'.SYSIN.VTOC'
   address tso "delete '"P_sysin"'"
   Sysin = SYSDSN("'"P_sysin"'")
   if Sysin /= "OK" then do
     address tso "ALLOCATE DDNAME(SYSIN) NEW UNIT(SYSDA) SPACE(1,1)",
            "TRACKS REUSE DSNAME('"P_sysin"')"
   end
   else do
     SAY  '*** Error *** 'P_sysin Sysin
     exit(8)
   end
   sin.1 = " LISTVTOC FORMAT,VOL=339Ø="||P_volnam
   "execio * diskw SYSIN (FINIS stem sin. "
RETURN
/*   */
/*   */
P2ØØØ_Allocate_Output:

address tso "delete '"||PREFX||"."||USERID()||".SYSPRINT'"
address tso "delete '"||PREFX||"."||USERID()||".TEMPVT'"
address tso "ALLOC DDNAME(SYSPRINT) NEW UNIT(SYSDA) space(2,2)",
          " cyl reuse  DSNAME('"||PREFX||"."||USERID()||".SYSPRINT')"

address tso "ALLOC F(DDNAME1) NEW UNIT(339Ø) VOLUME("||P_volnam||")",
            "tracks SPACE(1,1) DSNAME('HRDBA."||USERID()||".TEMPVT')"

Return
/*   */
/*   */
P3ØØØ_Execute_IEHLIST:

 address tso "IEHLIST "

"execio * DISKR SYSPRINT (FINIS STEM prt."
last = prt.Ø
linreq = last-2
out.1 = prt.linreq
say last out.1

"execio * DISKW SYSPRINT (FINIS STEM out."

/*  comment the signal code below to browse SYSPRINT dataset */
/*  this may be done for debugging                           */
```

```
   signal  TEMPSTEP

/* BROWSE THE SYSPRINT FILE */
ADDRESS ISPEXEC "LMINIT DATAID(DSID) DDNAME(SYSPRINT)"
ADDRESS ISPEXEC "BROWSE DATAID("DSID")"
ADDRESS ISPEXEC "LMFREE DATAID("DSID")"

TEMPSTEP:

/* address tso "STEPLIB FREE"  */

RETURN
/*   */
P4ØØØ_Clean_up:
address tso "FREE DDNAME(SYSPRINT)"
address tso "FREE DDNAME(SYSIN)"
address tso "FREE DDNAME(DDNAME1)"
address tso "delete '"||PREFX||"."||USERID()||".TEMPVT'"
address tso "delete '"P_sysin"'"
RETURN
/*   */
```

Jaiwant K Jonathan
DB2 DBA (USA)

DB2 PLAN_TABLE – access and maintenance

DB2 stores SQL access path-related details for each program in a user-supplied table called PLAN_TABLE. This table must exist for any bind process with the EXPLAIN option equal to YES. The information in the PLAN_TABLE is used in designing tables and indexes, and helps in the performance tuning of SQL queries used in application programs. A PLAN_TABLE in a production environment contains EXPLAIN results for all programs moved to the production environment, and, therefore, this data must be maintained in the same way as other production data. This article describes how accesses to a PLAN_TABLE can be made more efficiently, resulting in CPU and elapsed time savings. The article also provides some useful tips on how to maintain a PLAN_TABLE in a production environment.

EXPLAIN AND PLAN_TABLE

EXPLAIN is a monitoring tool that produces information about a plan, package, or SQL statement when it is bound. The output from EXPLAIN appears in a table called a PLAN_TABLE. The information in a PLAN_TABLE helps in determining the access path chosen for a query, designing databases, indexes, and application programs, and determining when to rebind an application.

POPULATING A PLAN_TABLE

Mostly, a PLAN_TABLE has rows inserted into it when a plan or a package is bound or rebound with the option ‘EXPLAIN (YES)’. EXPLAIN obtains information about the access paths for all explainable SQL statements in a package or the DBRMs of a plan. This information gets created in <package_owner>.PLAN_TABLE or <plan_owner>.PLAN_TABLE.

We can also populate a PLAN_TABLE by executing the SQL statement EXPLAIN and by specifying a single explainable SQL statement in the FOR clause as shown in Example 1 below. The resulting rows are created in <current_sql_id>.PLAN_TABLE.

Example 1

```
EXPLAIN PLAN
SET QUERYNO = 20 FOR
< explainable_select statement >
```

RETRIEVING ROWS FROM PLAN_TABLE

There are several processes that can insert rows in a PLAN_TABLE. In order to understand access paths, we must retrieve rows for a particular query in an appropriate order. All rows for a particular plan are identified by the value of APPLNAME or PROGNAME. All rows for a particular package are identified by the values of PROGNAME and COLLID (with no package versioning).

In order to retrieve this data for a particular package or a plan from PLAN_TABLE, a select statement like Example 2 is used – see below. Please note that the WHERE clause predicates are for PROGNAME and COLLID. If it was a package bind, we get EXPLAIN rows for that package because each package (no package versioning) can be uniquely identified by a collection-id and program name. If it was a plan bind, APPLNAME and PROGNAME will have the same value – that of the plan name – and the COLLID column is blank. Example 2 can be used to retrieve EXPLAIN rows for that particular plan.

Since the PLAN_TABLE does not have an index, any query on this table always results in a tablespace scan. If we run EXPLAIN for the SELECT statement in Example 2A, the output results (shown in Table 1) display a tablespace scan followed by an ORDERBY sort. As more and more programs are moved into production and because the same program is bound multiple times in production, this PLAN_TABLE also grows. Therefore any SELECT on this table will cost more and more CPU time as well as elapsed time. Also, if we wish to delete rows for a particular plan or package from this table, this deletion will also cause a tablespace scan, incurring a higher CPU cost.

Similarly, in order to retrieve EXPLAIN rows for a particular SQL statement that were inserted by EXPLAIN Example 1 (above) into the PLAN_TABLE, a SQL statement like Example 3 is used. The

EXPLAIN output for Example 3 (shown in Table 2) also verifies the tablespace scan followed by an ORDERBY sort.

Example 2

```
SELECT *
  FROM  SYSADM2.PLAN_TABLE
  WHERE PROGNAME  =  'PROG1'
    AND COLLID    =  'COLLECTION1'
    AND TIMESTAMP >  '1998011508000000'
  ORDER BY
        QUERYNO,
        QBLOCKNO,
        PLANNO,
        MIXOPSEQ
```

Table 1

EXPLAIN results for Example 2:

```
QUERYNO QBLOCKNO PROGNAME PLANNO METHOD CREATOR TNAME     ACCESSTYPE MATCHCOLS

    10        1  DSNESM68     1      0  SYSADM2 PLN_TBLE         R         0
    10        1  DSNESM68     2      3                                     0

ACCESSCREATOR  ACCESSNAME  INDEXONLY  SORTC_ORDERBY  COLLID

                           N          N              DSNESPCS
                           N          Y              DSNESPCS
```

Example 2A

```
SELECT *
  FROM  SYSADM2.PLAN_TABLE
  WHERE PROGNAME  =  'PROG1'
    AND COLLID    =  'COLLECTION1'
    AND TIMESTAMP >  '1998011508000000'
  ORDER BY
        PROGNAME,
        COLLID,
        QUERYNO,
        QBLOCKNO,
        PLANNO,
        MIXOPSEQ
```

Example 3

```
SELECT *
  FROM  SYSADM2.PLAN_TABLE
  WHERE QUERYNO  =  2Ø
  ORDER BY
        QBLOCKNO,
        PLANNO,
        MIXOPSEQ
```

Table 2

EXPLAIN results for Example 3:

```
QUERYNO QBLOCKNO PROGNAME PLANNO METHOD CREATOR TNAME     ACCESSTYPE MATCHCOLS

    2Ø        1  DSNESM68     1      Ø  SYSADM2 PLN_TBLE         R         Ø
    2Ø        1  DSNESM68     2      3                                     Ø

ACCESSCREATOR  ACCESSNAME  INDEXONLY  SORTC_ORDERBY  COLLID

                           N          N              DSNESPCS
                           N          Y              DSNESPCS
```

INDEX ON PLAN_TABLE

This tablespace scan can be avoided by creating a clustering index, say PLAN_INDEX, with keys as described in the following SQL statement:

```
CREATE INDEX SYSADM2.PLAN_INDEX
          ON SYSADM2.PLAN_TABLE
       (PROGNAME ,
        COLLID ,
        QUERYNO ,
        QBLOCKNO ,
        PLANNO ,
        MIXOPSEQ,
        TIMESTAMP )
       USING VCAT VCAT1
       FREEPAGE Ø PCTFREE  5
       CLUSTER
```

With this index, Example 2, which retrieves output rows for a particular plan/package, can now be modified to Example 2A (above) and the EXPLAIN results for statement 2A are described in Table 3 (also below). We can see that the access path has now improved to an index scan with MATCHCOLS = 2 and does not have an ORDERBY sort. This definitely results in less CPU and elapsed time as compared to the example without an index. In a PLAN_TABLE containing 190,000 rows, retrieving 7 rows for a program, use of an index (see

```
Statement 2         Statement 2A    Statement 3         Statement 3A
(without index)     (with index)    (without index)     (with index)

   1.24                 Ø.22             1.17                Ø.19
```

Figure 1: CPU times (in seconds)

Figure 1) brings down the CPU time to 0.22 seconds as compared to 1.24 seconds of CPU without using an index. These output rows for a plan or package can later be deleted from the PLAN_TABLE with SQL Example 4, which involves an index scan with MATCHCOLS = 2.

If the EXPLAIN rows were created for a particular SQL statement by using the SQL statement EXPLAIN as in Example 1, then these rows can be retrieved by changing Example 3 to Example 3A in order to include PROGNAME and COLLID in the 'where' clause and 'order by' clause as shown in Example 3A. EXPLAIN results for statement 3A are shown in Table 4, depicting improvements with an index scan with MATCHCOLS = 3 and with no ORDERBY sort. In a PLAN_TABLE containing 190,000 rows, retrieving 4 rows for a query, using an index (see Figure 1) brings down CPU time to 0.19 seconds as compared to 1.17 seconds of CPU without using an index.

Please note that, here, PROGNAME and COLLID correspond to the DYNAMIC SQL program used for running the SQL statement EXPLAIN. It can be a SPUFI, QMF, or DSNTEP2, or any other program depending on which one was used to create the rows in the PLAN_TABLE. (We can also find out values for PROGNAME and COLLID as a one-time exercise by browsing through PLAN_TABLE rows for column QUERYNO equal to the value set for QUERYNO during execution of the SQL statement EXPLAIN.) In our example, we have COLLID = DSNESPCS and PROGNAME = DSNESM68. These EXPLAIN rows can later be deleted by Example 5, which involves an index scan with MATCHCOLS = 3 .

In all cases, by introducing an index on a PLAN_TABLE, we can see significant improvements in CPU and elapsed times.

Table 3

EXPLAIN results for Statement 2A (with an index on the PLAN_TABLE):

```
QUERYNO QBLOCKNO PROGNAME PLANNO METHOD CREATOR TNAME     ACCESSTYPE MATCHCOLS

    3Ø        1  DSNESM68     1      Ø  SYSADM2 PLN_TBLE       I          2

ACCESSCREATOR  ACCESSNAME  INDEXONLY  SORTC_ORDERBY  COLLID

SYSADM2        PLAN_INDEX  N          N              DSNESPCS
```

Example 3A

```
SELECT *
  FROM  SYSADM2.PLAN_TABLE
  WHERE PROGNAME  =  'DSNESM68'
    AND COLLID    =  'DSNESPCS'
    AND QUERYNO   =  4Ø
  ORDER BY
        PROGNAME,
        COLLID,
        QUERYNO,
        QBLOCKNO,
        PLANNO,
        MIXOPSEQ
```

Table 4

EXPLAIN results for Statement 3A (with an index on the PLAN_TABLE):

```
QUERYNO QBLOCKNO PROGNAME PLANNO METHOD CREATOR TNAME     ACCESSTYPE MATCHCOLS

    4Ø        1  DSNESM68     1      Ø  SYSADM2 PLN_TBLE       I          3

ACCESSCREATOR  ACCESSNAME  INDEXONLY  SORTC_ORDERBY  COLLID

SYSADM2        PLAN_INDEX  N          N              DSNESPCS
```

Example 4

```
DELETE
  FROM  SYSADM2.PLAN_TABLE
  WHERE PROGNAME  =  'PROG1'
    AND COLLID    =  'COLLECTION1'
    AND TIMESTAMP >  '1998Ø115Ø8ØØØØØØ'
```

Example 5

```
DELETE
  FROM  SYSADM2.PLAN_TABLE
  WHERE PROGNAME  =  'DSNESM68'
    AND COLLID    =  'DSNESPCS'
    AND QUERYNO   =  40
```

MAINTAINING EXPLAIN RESULTS FOR MULTIPLE BINDS AND REBINDS

Because the same program is bound and rebound many times in production, these EXPLAIN output rows keep on accumulating in the PLAN_TABLE with different values for the timestamp. As a policy, we must maintain EXPLAIN output data for the earlier versions (at least the previous version) of each program in a PLAN_TABLE. This helps when comparing the access paths of the current version of a program with that of the earlier one, observing differences if any, analysing reasons for those differences, and taking appropriate action. This may be needed when a program performs badly after a rebind (because of a change in catalog statistics) or when a new version of a program is moved to production and it starts performing poorly.

In a PLAN_TABLE, each set of rows for a particular version of a program can be identified only by a range of timestamp values. Therefore, retrieval of these rows for the current version of a program, or for any earlier version, becomes very difficult because we have to first find out the timestamp values for the first and last row in the set before we can execute the actual SELECT statement for retrieval of rows.

PLAN HISTORY TABLE

One of the alternative ways to resolve this issue is to create a plan history table called PLAN_HIST_TABLE, which is a mirror image of the PLAN_TABLE. Since most of the time we are concerned with EXPLAIN results only for current versions of programs in production, it is a good idea to transfer all existing rows for a program from the PLAN_TABLE to a plan history table and delete those rows in the PLAN_TABLE before we perform a bind or rebind for that program.

The steps to be taken during each bind /rebind of a plan or a package are shown below:

- Step 1:

```
INSERT INTO PLAN_HIST_TABLE
         SELECT *
         FROM PLAN_TABLE
         WHERE PROGNAME  =  'PROG1'
        AND COLLID    =  'COLLECTION1'
```

- Step 2:

```
   DELETE FROM PLAN_TABLE
          WHERE PROGNAME  =  'PROG1'
         AND COLLID    =  'COLLECTION1'
```

- Step 3 – bind/rebind package or plan.

By doing this, we can maintain data for all earlier versions of a program in a plan history table and a PLAN_TABLE will contain data only for current versions of all programs. Because the plan history table will have many more rows than the PLAN_TABLE, we need to build an index (similar to PLAN_INDEX) on the PLAN_HIST_ TABLE, which could be called PLAN_HIST_INDEX, and which will provide efficient access for any query made on the plan history table.

As this history table grows with time, we must plan to delete rows for older versions of the program that may not be of any importance for the installation. We may also like to have a policy to keep up to the last two versions of a program in this table.

PLAN TABLE AND PLAN HISTORY TABLE MAINTENANCE

As we can see, there are frequent inserts and deletes in a PLAN_TABLE and plan history table. Therefore, these tables become candidates for periodic reorganization (the frequency of reorganization will depend on how many binds take place in a day, week, or month, and when the rows are deleted from these tables). A RUNSTATS taken at the appropriate time on these tables will ensure improved access on them.

Also, these tables contain a very important source of information that is required for the performance tuning of programs, making database changes to improve access paths, etc. We must take image copies of these tables regularly, like any other production tables.

CONCLUSION

PLAN_TABLE data in production is a very important source of information for making changes to application programs, database designs, and performance tuning. In order to avoid any data loss, this table should be regularly image-copied. Also, this table is volatile and therefore it must be re-organized periodically in order to maintain efficient access to any query on this table.

Sharad Kumar Pande
Senior DB2 DBA
PricewaterhouseCoopers (USA)

Image copy, DSNTIAUL copy, and disaster recovery of DB2 objects – part 2

This month we conclude with the code that makes it simpler to prepare image copy jobs for tablespaces and DSNTIAUL copy jobs for tables that require GDGs to be defined, and to write JCL for each object created.

```
WRITE_SYSIN:PROCEDURE EXPOSE UPPERL LOWERL TSPARTCNT. TSNAMES.
  DO MAIN_LP=UPPERL TO LOWERL BY -1
    TSNAME=STRIP(SUBSTR(TSNAMES.MAIN_LP,1,8),'B') || '.' || ,
           STRIP(SUBSTR(TSNAMES.MAIN_LP,9,8),'B')
    IF TSPARTCNT.MAIN_LP=Ø THEN DO
      LINE = '   COPY TABLESPACE ' || TSNAME || ,
             ' COPYDDN S'||MAIN_LP||' SHRLEVEL CHANGE'
      PUSH LINE
    END
    ELSE DO
      DO PARTNUM=TSPARTCNT.MAIN_LP TO 1 BY -1
        DD_NAME=TRANSLATE(FORMAT(MAIN_LP,4),'Ø',' ')||,
                TRANSLATE(FORMAT(PARTNUM,3),'Ø',' ');
        LINE = '       COPYDDN S'||DD_NAME||' SHRLEVEL CHANGE'
        PUSH LINE
        LINE = '   COPY TABLESPACE ' || TSNAME || ' DSNUM ' || PARTNUM
        PUSH LINE
      END
    END
  END
  PUSH '//DSNUPROC.SYSIN DD *'
  £EXECIO * DISKW ICJCL£
RETURN;
JOB_CARD_ICOPY:PROCEDURE EXPOSE DB2ID JC JOB_NUM
  JOB_NUM=JOB_NUM+1
  PUSH ''
  LINE='//ICOPY'||JOB_NUM||' EXEC DSNUPROC,SYSTEM=DB'||DB2ID||'Ø' ,
            ',UID=''ICPY'||DB2ID||JOB_NUM||''',UTPROC='''''
  PUSH LINE
  PUSH '// MSGLEVEL=(1,1),REGION=ØM,NOTIFY=SKMXSYP,TYPRUN=HOLD'
  LINE='//ICOPY'||JOB_NUM||'  JOB (ACCT#),'IMAGECOPY','||,
       'MSGCLASS=X,CLASS='||JC||','
  PUSH LINE
  £EXECIO * DISKW ICJCL£
RETURN
JOB_CARD_UNLD:PROCEDURE EXPOSE DB2ID JC
  PUSH ''
  LINE='//         DD DISP=SHR,DSN='||DB2ID||'DSN.SDSNEXIT'
  PUSH LINE
  LINE='//JOBLIB   DD DISP=SHR,DSN='||DB2ID||'DSN.SDSNLOAD'
  PUSH LINE
  PUSH '// MSGLEVEL=(1,1),REGION=ØM,NOTIFY=SKMXSYP,TYPRUN=HOLD'
  PUSH '//UNLOAD   JOB (ACCT#),'UNLOAD',MSGCLASS=X,CLASS='||JC||','
  £EXECIO * DISKW UNJCL£
RETURN
UNLOAD_SYSIN:PROCEDURE EXPOSE TBNAMES. REMAINDER UN_IN ,
                     DB2ID FROM_TAB TO_TAB JCLEXT OLDJCLEXT
  OLDJCLEXT=JCLEXT
  DO WHILE ( OLDJCLEXT = JCLEXT )
    JCLEXT=TIME(S)
  END
  PUSH ''
  PUSH '/*'
  DO MAIN_LP=FROM_TAB TO TO_TAB BY -1
    TBNAME=STRIP(SUBSTR(TBNAMES.MAIN_LP,1,8),'B') || '.' || ,
         TRANSLATE(STRIP(SUBSTR(TBNAMES.MAIN_LP,11,18),'B'),' ','ØØ'X)
    LINE = '   SELECT * FROM '||TBNAME||' WITH UR;'
    PUSH LINE
  END
  PUSH '//SYSIN    DD *'
  PUSH '//       DISP=(NEW,DELETE,DELETE),SPACE=(8ØØ,(29Ø,9Ø),RLSE)'
  LINE='//SYSPUNCH DD DSN=SYSPDBA.PSØ.UNLOAD.JCL'||JCLEXT||'.TEMP,'
  PUSH LINE
  £EXECIO * DISKW UNJCL£
RETURN
NEW_STEP:PROCEDURE EXPOSE INT_PART DB2ID
  PUSH ''
  PUSH '//SYSUDUMP DD SYSOUT=*'
  PUSH '//SYSPRINT DD SYSOUT=*'
  LINE='      LIB('''||DB2ID||'DSN.RUNLIB.LOAD'')'
  PUSH LINE
  PUSH ' RUN  PROGRAM(DSNTIAUL) PLAN(DSNTIAUL) PARMS(''SQL'') - '
  LINE=' DSN SYSTEM(DB'||DB2ID||'Ø)'
  PUSH LINE
  PUSH '//SYSTSIN  DD *'
  PUSH '//SYSTSPRT DD SYSOUT=*'
  LINE='//UNLD'||INT_PART||' EXEC PGM=IKJEFTØ1,REGION=ØM'
  PUSH LINE
  £EXECIO * DISKW UNJCL£
RETURN
ADD_DD_COPY:PROCEDURE EXPOSE MAIN_LP TSNAME DB2ID ,
                RETPD_COPY PARTNUM PREV_DD LABEL_NUM TSPARTCNT.
  LABEL_NUM=LABEL_NUM+1;
  IF TSPARTCNT.MAIN_LP > Ø THEN DO
    DSNAME='ODM.GBØ.S1D.'||TSNAME||'.P'||PARTNUM||'(+1)'
    DD_NAME=TRANSLATE(FORMAT(MAIN_LP,4),'Ø',' ')||,
            TRANSLATE(FORMAT(PARTNUM,3),'Ø',' ');
  END
  ELSE DO
    DSNAME='ODM.GBØ.S1D.'||TSNAME||'(+1)'
    DD_NAME=TRANSLATE(FORMAT(MAIN_LP,4),'Ø',' ');
  END;
  PUSH ''
  IF MAIN_LP = 1 & PARTNUM=1 | LABEL_NUM=1 THEN DO
    LINE='// DCB=BLKSIZE=3276Ø,TRTCH=COMP,BUFNO=2Ø,RETPD='||RETPD_COPY
    PUSH LINE
PUSH '// VOL=(,,,2Ø),DISP=(NEW,CATLG,CATLG),UNIT=CARTM,LABEL=(1,SL),'
    LINE='//S'||DD_NAME||' DD DSN='||DSNAME||','
    PUSH LINE
  END
  ELSE DO
    LINE='// DCB=BLKSIZE=3276Ø,TRTCH=COMP,BUFNO=2Ø,RETPD='||RETPD_COPY
    PUSH LINE
    LINE='// VOL=(,RETAIN,,2Ø,REF=*.S'||PREV_DD||'),'
    PUSH LINE
    LINE = '// DISP=(NEW,CATLG,CATLG),UNIT=AFF=S'||PREV_DD|| ,
           ',LABEL=('||LABEL_NUM||',SL),'
    PUSH LINE
    LINE='//S'||DD_NAME||' DD DSN='||DSNAME||','
    PUSH LINE
  END
  PREV_DD=DD_NAME;

  £EXECIO * DISKW ICJCL£
RETURN
ADD_DD_UNLD:PROCEDURE EXPOSE INT_PART REMAINDER TBNAME ,
           LAST_DD_NAME MAIN_LP UNL_DS_NAME UN_IN DB2ID RETPD_UNLD
  DD_ID=REMAINDER
  IF REMAINDER = Ø THEN DD_ID=UN_IN
  CURRENT_PTR=TRANSLATE(FORMAT(DD_ID - 1,2),'Ø',' ')
  PREV_PTR=TRANSLATE(FORMAT(DD_ID - 2,2),'Ø',' ')
  PUSH ''
  IF DD_ID = 1 THEN DO
    LINE='// DCB=BLKSIZE=1Ø24,TRTCH=COMP,BUFNO=2Ø,RETPD='||RETPD_UNLD
    PUSH LINE
    IF INT_PART=Ø THEN ,
 PUSH '// VOL=(,,,2Ø),DISP=(NEW,CATLG,CATLG),UNIT=CARTM,LABEL=(1,SL),'
    ELSE DO
      LINE='// VOL=(,RETAIN,,2Ø,REF=*.LASTDD),'
      PUSH LINE
      LINE = '// DISP=(NEW,CATLG,CATLG),UNIT=AFF=LASTDD'|| ,
             ',LABEL=('||MAIN_LP||',SL),'
      PUSH LINE
    END
    LINE='//SYSRECØØ DD DSN=UNLOAD.GBØ.'|| ,
         UNL_DS_NAME||'(+1)'||','
    PUSH LINE
    IF INT_PART<>Ø THEN DO
      LINE='//LASTDD   DD DSN='||LAST_DD_NAME||',DISP=OLD'
      PUSH LINE
    END
  END
  ELSE DO
    LINE='// VOL=(,RETAIN,,2Ø,REF=*.SYSREC'||PREV_PTR||')'
    PUSH LINE
LINE='// DCB=BLKSIZE=3276Ø,TRTCH=COMP,BUFNO=2Ø,RETPD='||RETPD_UNLD||','
    PUSH LINE
    LINE = '// DISP=(NEW,CATLG,CATLG),UNIT=AFF=SYSREC'||PREV_PTR|| ,
           ',LABEL=('||MAIN_LP||',SL),'
    PUSH LINE
    LINE='//SYSREC'||CURRENT_PTR||' DD DSN=UNLOAD.GBØ.'|| ,
         UNL_DS_NAME||'(+1)'||',' ;
    PUSH LINE
    LAST_DD_NAME='UNLOAD.GBØ.'||,
                 UNL_DS_NAME||'(+1)';
  END
  £EXECIO * DISKW UNJCL£
RETURN
DO_SELECT:PROCEDURE EXPOSE SEL. DB2ID
  PUSH ''
  DO X = SEL.Ø TO 1 BY -1
    PUSH SEL.X
  END

  £EXECIO * DISKW SYSIN (FINIS£
  CMD = £RUN PROGRAM(DSNTIAUL) PLAN(DSNTIAUL)£
  CMD = CMD || ' LIB('''||DB2ID||'DSN.RUNLIB.LOAD'')'
  CMD = CMD || £ PARMS('SQL')£
  QUEUE 'END '
  IF DB2ID='D' THEN 'DSN SYSTEM(DBDØ)'
  IF DB2ID='T' THEN 'DSN SYSTEM(DBTØ)'
  IF DB2ID='E' THEN 'DSN SYSTEM(DBEØ)'
  IF DB2ID='P' THEN 'DSN SYSTEM(DBPØ)'
  IF RC > Ø THEN DO
       SAY 'CAN NOT CONNECT TO DB2 SUBSYSTEM.'
       SAY 'PLEASE TRY LATER...'
       RETURN
  END
  QUEUE CMD
  QUEUE 'END '
  IF DB2ID='D' THEN 'DSN SYSTEM(DBDØ)'
  IF DB2ID='T' THEN 'DSN SYSTEM(DBTØ)'
  IF DB2ID='E' THEN 'DSN SYSTEM(DBEØ)'
  IF DB2ID='P' THEN 'DSN SYSTEM(DBPØ)'
  £EXECIO * DISKR SYSPRINT (STEM SQLHATA.£
  UNLD_OK=Ø
  DO SQ_LP=1 TO SQLHATA.Ø
    IF INDEX(SQLHATA.SQ_LP,'DSNT495I SUCCESSFUL UNLOAD') > Ø THEN ,
      UNLD_OK=1
  END
  IF UNLD_OK=Ø THEN DO
    SAY 'THERE IS AN ERROR IN SQL STATEMENT.'
    DO SQ_LP2=1 TO SQLHATA.Ø
      SAY SQLHATA.SQ_LP2
    END
    EXIT 2Ø
  END
  PUSH ''
  £EXECIO * DISKW SYSPRINT (FINIS£
RETURN
ADD_WTO:PROCEDURE EXPOSE OPR_TYPE
  PUSH ''
  PUSH '//'
  PUSH '/*'
  IF OPR_TYPE = 'UNLOAD' THEN DO
    PUSH '->****************************************************'
    PUSH '->SYS2.BACKUP.JCLLIB(UNLRES) JCL.'
    PUSH '->SYS2.OPERLIB(UNLRESTB) AND RESTART THE JOB WITH '
    PUSH '->TABLE NAME DUMPED SUCCESSFULLY TO THE DATASET '
    PUSH '->UNLOAD JOB ENDED WITH ERROR. PLEASE WRITE THE LAST '
    PUSH '->****************************************************'
  END
  ELSE DO
    PUSH '-> ***************************************************'
    PUSH '-> ARE COMPLETED.'
    PUSH '-> SYS2.BACKUP.JCLLIB(ICOPYX) AFTER ALL ICOPYX JOBS'
    PUSH '-> TABLESPACE, RESTART THE JOB WITH THE JCL '
    PUSH '-> ERROR. WHEN YOU ARE SURE THAT THERE IS NO RESTRICTED'
    PUSH '-> ICOPY JOB ENDED WITH ERROR. PLEASE INVESTIGATE THE '
    PUSH '-> ***************************************************'
  END
  PUSH '//SYSIN    DD   *'
  PUSH '//ERROR    EXEC IPOWTO,REGION=768K,COND=((4,GE),EVEN)'
  IF OPR_TYPE = 'UNLOAD' THEN £EXECIO * DISKW UNJCL£
  IF OPR_TYPE = 'COPY' THEN £EXECIO * DISKW ICJCL£
RETURN
```

ICOPYX

```
//ICOPYX  JOB (ACCT#),'',MSGCLASS=X,CLASS=9,
// MSGLEVEL=(1,1),REGION=4M
//*************************************************************
//* ICOPYALL DB2ID OP_ID UNLINT COPY_JCL_NAME UNLOAD_JCL_NAME
//*   RETPD_COPY RETPD_UNLD TS_SELECT TB_SELECT PARTITION JOB_CLASS
//*
//* DB2ID  : DB2 SUBSYSTEM ID. IT MAY BE D, T, E, OR P
//*
//* OP_ID  : MAY BE UNLD , COPY OR BOTH.
//*          UNLD : ONLY DSNTIAUL COPY.
//*          COPY : ONLY IMAGE COPY.
//*          BOTH : BOTH OF DSNTIAUL COPY AND IMAGE COPY.
//*
//* UNLINT          : CREATES A NEW STEP EVERY UNLINT DD.
//* COPY_JCL_NAME   : IMAGE COPY JCL TO BE CREATED.
//* UNLOAD_JCL_NAME : DSNTIAUL JCL TO BE CREATED.
//* RETPD_COPY      : RETENTION PERIOD OF IMAGE COPY DATASETS.
//* RETPD_UNLD      : RETENTION PERIOD OF DSNTIAUL COPY DATASETS.
//* TS_SELECT       : QUERY THAT SELECTS TABLESPACES TO BE IMAGE
//*                   COPIED.
//* TB_SELECT       : QUERY THAT SELECTS TABLES TO BE COPIED
//*                   USING DSNTIAUL.
//* JOB_CLASS       : JOB CLASS OF THE JCLS.
//* JOB_COUNT       : JOB COUNT.
//*                   IMAGE COPIES WILL BE DIVIDED INTO THIS NUMBER.
//* PARTITIONED     : 'YES' IF IMAGE COPIES WILL BE TAKEN PARTITIONED
//*                   TABLESPACE LEVEL, OTHERWISE 'NO'.
//* IS_RESTART      : MUST BE CODED 'RESTART' , IF DSTIAUL COPY
//*                   IS TO BE RESTARTED.
//*****************************************************************
//RUNEXEC  EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø,REGION=8192K
//STEPLIB  DD DSN=ISP.SISPLOAD,DISP=SHR
//         DD DSN=PDSN.SDSNLOAD,DISP=SHR
//SYSEXEC  DD   DSN=SYSPDBA.REXXLIB,DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSRESIN DD DSN=SYS2.OPERLIB(UNLRESTB),DISP=SHR
//SYSTSIN  DD   *
 EXECUTIL SEARCHDD(YES)
 %ICOPYALL P COPY 9Ø PCOPYUPD PUNLDUPD 7 8 -
  TSSELUPD TBSELALL 9 4 NO
/*
```

RECALL

```
/*   REXX   */
 PARSE ARG DB2ID OP_ID RECTS_DSNAME REC_IXDSNAME JOB_COUNT PARTITION
 STEP_CNT=Ø
 CNT=Ø
 G_STEP_CNT=Ø
 JOB_CNT=1
 /* MAX KILOBYTES AFTER THAT WE CATALOG WORK SPACES */
 MAX_KB_WORK_CATLG=4ØØØØ;
```

```
/* MAX KILOBYTES THAT A DASD CAN ALLOCATE */
MAX_KB_WORK=2600000
TARIH = DATE('E')
SAAT = TIME()
£ALLOC FI(SYSPRINT) RECFM(F B) LRECL(133) SPACE(1,1) BLOCK(4096)£
£ALLOC FI(SYSPUNCH) RECFM(F B) LRECL(80) SPACE(1,1) BLOCK(4096)£
£ALLOC FI(SYSIN) RECFM(F B) LRECL(80) BLKSIZE(80)£
£ALLOC FI(RETSJCL) DA('SYS2.BACKUP.JCLLIB(£||RECTS_DSNAME||£)') SHR£
£ALLOC FI(LISTCATO) RECFM(V B) LRECL(125) SPACE(1,1) BLOCK(629)£
CALL JOB_CARD_RECOVER_TS
SEL.0=9
SEL.1=' SELECT DBNAME,NAME'
SEL.2=' FROM SYSIBM.SYSTABLESPACE '
SEL.3=' WHERE   DBNAME NOT LIKE '||''''||'WRK%'||''''
SEL.4='     AND NAME NOT LIKE   '||''''||'UNLOAD%'||''''
SEL.5='     AND NAME NOT LIKE   '||''''||'_CPY%'||''''
SEL.6='     AND DBNAME <> '||''''||'BMCARM'||''''
SEL.7='     AND DBNAME NOT LIKE '||''''||'DSN%'||''''
SEL.8='     AND DBNAME NOT LIKE '||''''||'DSQ%'||''''
SEL.9='     ORDER BY DBNAME,NAME;'
£ALLOC FI(SYSREC00) SPACE(1,1) BLOCK(4096)£
CALL DO_SELECT
£EXECIO * DISKR SYSREC00 (STEM TSNAMES.£
£EXECIO * DISKR SYSREC00 (FINIS£
£FREE FI(SYSREC00)£
REC_CNT=0;
DO MAIN_LP=1 TO TSNAMES.0
  SAY MAIN_LP
  DBNAME=STRIP(SUBSTR(TSNAMES.MAIN_LP,1,8),'B')
  TSNAME=STRIP(SUBSTR(TSNAMES.MAIN_LP,9,8),'B')
  IF PARTITION='YES' THEN DO
    SEL.0=3;
    SEL.1=' SELECT CHAR(DECIMAL(PARTITIONS)) '
    SEL.2=' FROM SYSIBM.SYSTABLESPACE '
    SEL.3=' WHERE NAME='||''''||TSNAME''''||';'
    £ALLOC FI(SYSREC00) SPACE(1,1) BLOCK(4096)£
    CALL DO_SELECT
    £EXECIO * DISKR SYSREC00 (STEM PARTCNT.£
    £EXECIO * DISKR SYSREC00 (FINIS£
    £FREE FI(SYSREC00)£
    PARTCNT.1=SUBSTR(PARTCNT.1,2,5);
    PARTCNT.1=STRIP(TRANSLATE(PARTCNT.1,' ','00'X),'B');
    DO WHILE ( SUBSTR(PARTCNT.1,1,1) = '0' & LENGTH(PARTCNT.1) > 1 )
      IF SUBSTR(PARTCNT.1,1,1)='0' THEN ,
        PARTCNT.1=SUBSTR(PARTCNT.1,2,LENGTH(PARTCNT.1)-1)
    END;
    TSPARTCNT=PARTCNT.1
  END
  ELSE TSPARTCNT=0
```

```
    IF TSPARTCNT=Ø THEN PARTNUM_LAST=1
       ELSE PARTNUM_LAST=TSPARTCNT
    DO PARTNUM=1 TO PARTNUM_LAST
      REC_CNT=REC_CNT+1;
      VOLCNT.REC_CNT=Ø;
      IF TSPARTCNT=Ø THEN NUMPART.REC_CNT=Ø
         ELSE NUMPART.REC_CNT=PARTNUM
      DBNAME.REC_CNT=DBNAME
      TSNAME.REC_CNT=TSNAME
      SEL.Ø=7;
      SEL.1=' SELECT DSNAME,TIMESTAMP FROM SYSIBM.SYSCOPY'
      SEL.2=' WHERE DBNAME='||''''||DBNAME||''''||' AND'
      SEL.3='       TSNAME='||''''||TSNAME||''''||' AND'
      SEL.4='       DSNAME LIKE '||''''||'ODM%'||''''||' AND'
      IF TSPARTCNT=Ø THEN ,
        SEL.5='        DSNUM=Ø AND'
      ELSE ,
        SEL.5='        DSNUM='||PARTNUM||' AND'
      SEL.6='       ICTYPE='||''''||'F'||''''
      SEL.7=' ORDER BY TIMESTAMP DESC;'
      £ALLOC FI(SYSRECØØ) SPACE(1,1) BLOCK(4Ø96)£
      CALL DO_SELECT
      £EXECIO * DISKR SYSRECØØ (STEM DSNAMES.£
      £EXECIO * DISKR SYSRECØØ (FINIS£
      £FREE FI(SYSRECØØ)£
      LAST_COPY_DSN.REC_CNT=STRIP(SUBSTR(DSNAMES.1,1,44),'B');
      PROFILE NOPREFIX
      £LISTCAT ENT(£LAST_COPY_DSN.REC_CNT£) OFILE(LISTCATO) ALL£
      IF RC <> Ø THEN DO
        SAY 'AN ERROR IS ENCOUNTERED WHILE TAKING LISTCAT...'
        EXIT 2Ø
      END;
      £EXECIO * DISKR LISTCATO (STEM LISTC. £
      DO LC=1 TO LISTC.Ø
        IF INDEX(LISTC.LC,'VOLSER') > Ø THEN DO;
          VOLCNT.REC_CNT=VOLCNT.REC_CNT+1
          VOLSER.REC_CNT=SUBSTR(LISTC.LC,27,6)
        END;
      END;
      £EXECIO * DISKR LISTCATO (FINIS £
      SAY 'DB NAME=' DBNAME.REC_CNT 'TS NAME=' TSNAME.REC_CNT ,
          'VOLSER=' VOLSER.REC_CNT
      SAY 'LAST IC DSN NAME=' LAST_COPY_DSN.REC_CNT
      SAY '-----------------------------------------------------'
    END
  END
  /* WRITE JCL */
  PREV_VOLSER=''
  PREV_MAIN_LP=1
```

```
DO MAIN_LP=1 TO REC_CNT
  IF PREV_VOLSER <> VOLSER.MAIN_LP & PREV_VOLSER <> '' | ,
     VOLCNT.MAIN_LP > 1 THEN DO;
    FIRST_CNT=PREV_MAIN_LP
    LAST_CNT=MAIN_LP-1
    CALL WRITE_SYSIN
    CALL JOB_CARD_RECOVER_TS
    PREV_MAIN_LP=MAIN_LP
  END;
  FMLP=FORMAT(MAIN_LP,6)
  FMLP=TRANSLATE(FMLP,'Ø',' ')
  PUSH ''
  LINE='//             DISP=(OLD,PASS)'
  PUSH LINE
  IF MAIN_LP = PREV_MAIN_LP THEN ,
    LINE='//             UNIT=(,,DEFER),'
  ELSE ,
    LINE='//             UNIT=AFF=DD'||OLD_FMLP||','
  PUSH LINE
  LINE='//             VOL=(,RETAIN),'
  PUSH LINE
  LINE='//DD'||FMLP||' DD DSN='||LAST_COPY_DSN.MAIN_LP||','
  PUSH LINE
  OLD_FMLP=FMLP
  £EXECIO * DISKW RETSJCL£
  IF VOLCNT.MAIN_LP=1 THEN PREV_VOLSER=VOLSER.MAIN_LP
END
FIRST_CNT=PREV_MAIN_LP
LAST_CNT=MAIN_LP-1
CALL WRITE_SYSIN
PUSH ''
PUSH '//'
£EXECIO * DISKW RETSJCL (FINIS£
£FREE FI(RETSJCL)£
/*   INDEX RECOVER JCLS   */
£ALLOC FI(REIXJCL) DA('SYS2.BACKUP.JCLLIB(£||REC_IXDSNAME||£)') SHR£
CALL JOB_CARD_RECOVER_IX
SEL.Ø=25;
SEL.1=' SELECT IXCREATOR,IXNAME,'
SEL.2=' CHAR(DECIMAL((CARDF/1Ø24)*(TOTLEN+13),15)) FROM ('
SEL.3=' SELECT B.CREATOR AS IXCREATOR,'
SEL.4='        B.NAME AS IXNAME,A.CARDF AS CARDF,'
SEL.5='        SUM(LENGTH) AS TOTLEN'
SEL.6='   FROM SYSIBM.SYSTABLES A,SYSIBM.SYSINDEXES B,'
SEL.7='        SYSIBM.SYSKEYS C,SYSIBM.SYSCOLUMNS D,'
SEL.8='        SYSIBM.SYSTABLESPACE E'
SEL.9='   WHERE A.TSNAME=E.NAME AND '
SEL.1Ø='         A.NAME=B.TBNAME AND'
SEL.11='         A.CREATOR=B.TBCREATOR AND'
SEL.12='          A.DBNAME=E.DBNAME AND'
SEL.13='         B.NAME=C.IXNAME AND'
SEL.14='         C.COLNAME=D.NAME AND'
SEL.15='         A.NAME=D.TBNAME AND'
SEL.16='         A.CREATOR=D.TBCREATOR AND'
SEL.17='         B.CREATOR NOT LIKE '||''''||'SYSIBM%'||''''|| ' AND '
SEL.18='         B.CREATOR NOT LIKE '||''''||'UNL%'||''''|| ' AND '
SEL.19='         B.CREATOR NOT LIKE '||''''||'DSN%'||''''|| ' AND '
SEL.20='         E.DBNAME NOT LIKE '||''''||'DSN%'||''''|| ' AND '
SEL.21='         E.DBNAME NOT LIKE '||''''||'WRK%'||''''|| ' AND '
SEL.22='         E.NAME <> '||''''||'PCPYTAB'||''''|| ' AND '
SEL.23='         E.NAME NOT LIKE '||''''||'UNLOAD%'||''''|| ' AND '
SEL.24='         E.NAME NOT LIKE '||''''||'DSN%'||''''
SEL.25='  GROUP BY B.CREATOR,B.NAME,A.CARDF) AS QRY1;'
£ALLOC FI(SYSREC00) SPACE(1,1) BLOCK(4096)£
CALL DO_SELECT
£EXECIO * DISKR SYSREC00 (STEM SPACES.£
£EXECIO * DISKR SYSREC00 (FINIS£
£FREE FI(SYSREC00)£
HOWMANY_IX_PER_JOB=SPACES.0 % JOB_COUNT + 1
DO MAIN_LP=1 TO SPACES.0
  IX_CREATOR=SUBSTR(SPACES.MAIN_LP,1,8)
  IX_NAME=SUBSTR(SPACES.MAIN_LP,11,18)
  SPC_QTY=SUBSTR(SPACES.MAIN_LP,30,15)
  IX_CREATOR=STRIP(TRANSLATE(IX_CREATOR,' ','00'X),'B')
  IX_NAME=STRIP(TRANSLATE(IX_NAME,' ','00'X),'B')
  SPC_QTY=STRIP(TRANSLATE(SPC_QTY,' ','00'X),'B')
  /* IF STATISTICS HAS NOT BEEN COLLECTED, SUPPOSE IT IS 1000 */
  IF SUBSTR(SPC_QTY,1,1)='-' THEN DO
    SAY 'STATISTICS HAS NOT BEEN COLLECTED FOR INDEX:' IX_NAME
    SPACES.1=SUBSTR(SPC_QTY,2,LENGTH(SPC_QTY)-1)
    FAKTOR=1000
  END
  ELSE FAKTOR=1
  DO WHILE ( SUBSTR(SPC_QTY,1,1) = '0' & LENGTH(SPC_QTY) > 2 )
    IF SUBSTR(SPC_QTY,1,1)='0' THEN ,
      SPC_QTY=SUBSTR(SPC_QTY,2,LENGTH(SPC_QTY)-1)
  END;
  SPACE_FOR_IX_REC=SPC_QTY
  SPACE_FOR_IX_REC=SPACE_FOR_IX_REC*FAKTOR
  SAY 'INDEX NAME=' IX_CREATOR '.' IX_NAME 'SPACE=' SPACE_FOR_IX_REC
  SPACE_FOR_IX_REC1=SPACE_FOR_IX_REC % 5 +1
  IF SPACE_FOR_IX_REC1 > MAX_KB_WORK THEN ,
     SPACE_FOR_IX_REC1 = MAX_KB_WORK
  SPACE_FOR_IX_REC2=SPACE_FOR_IX_REC % 20 +1
  CALL JOB_STEP_RECOVER_IX
  LINE = '   RECOVER INDEX ('||IX_CREATOR||'.'||IX_NAME||')'
  PUSH LINE
  £EXECIO * DISKW REIXJCL£
  PUSH ''
  PUSH '/*'
  £EXECIO * DISKW REIXJCL£
  IF SPACE_FOR_IX_REC1 > MAX_KB_WORK_CATLG THEN DO
    STEP_CNT=STEP_CNT+1
    G_STEP_CNT=G_STEP_CNT+1
    PUSH ''
    PUSH '/*'
    PUSH '     DELETE SYSPDBA.PSØ.RECI'||JOB_CNT||'.SUT1.TEMP'
    PUSH '     DELETE SYSPDBA.PSØ.RECI'||JOB_CNT||'.WORK1.TEMP'
    PUSH '     DELETE SYSPDBA.PSØ.RECI'||JOB_CNT||'.WORK2.TEMP'
    PUSH '     DELETE SYSPDBA.PSØ.RECI'||JOB_CNT||'.WORK3.TEMP'
    PUSH '     DELETE SYSPDBA.PSØ.RECI'||JOB_CNT||'.WORK4.TEMP'
    PUSH '     DELETE SYSPDBA.PSØ.RECI'||JOB_CNT||'.WORK5.TEMP'
    PUSH '     DELETE SYSPDBA.PSØ.RECI'||JOB_CNT||'.WORK6.TEMP'
    PUSH '//SYSIN   DD *'
    PUSH '//SYSPRINT DD SYSOUT=*'
    LINE='//DELIX'||STEP_CNT||'  EXEC PGM=IDCAMS,COND=(4,GE,RECIX'||,
        STEP_CNT||')'
    PUSH LINE
    £EXECIO * DISKW REIXJCL£
  END
END
PUSH ''
PUSH '//'
£EXECIO * DISKW REIXJCL (FINIS£
£FREE FI(REIXJCL)£
STAT=MSG('OFF')
£FREE FI(SYSRECØØ)£
£FREE FI(SYSPRINT)£
£FREE FI(SYSPUNCH)£
£FREE FI(SYSIN)£
£FREE FI(TBSEL)£
£FREE FI(UNJCL)£
£FREE FI(LISTCATO)£
STAT=MSG('ON')
EXIT
JOB_CARD_RECOVER_TS:PROCEDURE EXPOSE DB2ID CNT TARIH SAAT
  CNT=CNT+1;
  PUSH ''
  PUSH '// MSGLEVEL=(1,1),REGION=ØM,NOTIFY=SKMXSYP,TYPRUN=HOLD'
  PUSH '//RECTS'||CNT||'  JOB (ACCT#),'REXTS',MSGCLASS=X,CLASS=9,'
  £EXECIO * DISKW RETSJCL£
  PUSH ''
  PUSH '//*       CREATED ON '||TARIH||'   '||SAAT
  £EXECIO * DISKW RETSJCL£
  CALL ADD_WTO;
  PUSH ''
  LINE='//RECOVER EXEC DSNUPROC,SYSTEM=DB'||DB2ID||'Ø'||,
             ',UID=''REC'||DB2ID||CNT||''',UTPROC='''''
  PUSH LINE
  £EXECIO * DISKW RETSJCL£
RETURN
JOB_STEP_RECOVER_IX:PROCEDURE EXPOSE DB2ID SPACE_FOR_IX_REC1 ,
                    SPACE_FOR_IX_REC2 ,
                    STEP_CNT G_STEP_CNT JOB_COUNT JOB_CNT ,
                    HOWMANY_IX_PER_JOB MAX_KB_WORK_CATLG ,
                    TARIH SAAT
  STEP_CNT=STEP_CNT+1
  G_STEP_CNT=G_STEP_CNT+1
  INT_PART = G_STEP_CNT % HOWMANY_IX_PER_JOB
  IF G_STEP_CNT=INT_PART*HOWMANY_IX_PER_JOB | ,
     SPACE_FOR_IX_REC1 > MAX_KB_WORK_CATLG | ,
     STEP_CNT > 2ØØ THEN DO;
    JOB_CNT=JOB_CNT+1
    STEP_CNT=1
    CALL JOB_CARD_RECOVER_IX
  END
  PUSH ''
  PUSH '//SYSIN   DD *'
  PUSH '//SYSPRINT DD SYSOUT=*'
  PUSH '//UTPRINT  DD SYSOUT=*'
  LINE='//            SPACE=(1Ø24,('||SPACE_FOR_IX_REC1||,
       ','||SPACE_FOR_IX_REC1||'),,,ROUND),VOL=(,,,2Ø)'
  PUSH LINE
  IF SPACE_FOR_IX_REC1 > MAX_KB_WORK_CATLG THEN ,
    LINE='//SYSUT1 DD DSN=SYSPDBA.PSØ.RECI'||JOB_CNT||,
          '.SUT1.TEMP,DISP=(NEW,CATLG,CATLG),'
  ELSE,
    LINE='//SYSUT1   DD DSN=SYSPDBA.PSØ.RECI'||JOB_CNT||,
          '.SUT1.TEMP,DISP=(NEW,DELETE,CATLG),'
  PUSH LINE
  IF SPACE_FOR_IX_REC1 > MAX_KB_WORK_CATLG THEN DO
    LINE='//            SPACE=(1Ø24,('||SPACE_FOR_IX_REC1||,
         ','||SPACE_FOR_IX_REC1||'),,,ROUND)'
    PUSH LINE
    LINE='//             DISP=(NEW,CATLG,CATLG),'
    PUSH LINE
    LINE='//SORTWKØ6 DD DSN=SYSPDBA.PSØ.RECI'||JOB_CNT||,
         '.WORK6.TEMP,'
    PUSH LINE
    LINE='//            SPACE=(1Ø24,('||SPACE_FOR_IX_REC1||,
         ','||SPACE_FOR_IX_REC1||'),,,ROUND)'
    PUSH LINE
    LINE='//            DISP=(NEW,CATLG,CATLG),'
    PUSH LINE
    LINE='//SORTWKØ5 DD DSN=SYSPDBA.PSØ.RECI'||JOB_CNT||,
         '.WORK5.TEMP,'
    PUSH LINE
    LINE='//            SPACE=(1Ø24,('||SPACE_FOR_IX_REC1||,
         ','||SPACE_FOR_IX_REC1||'),,,ROUND)'
    PUSH LINE
    LINE='//            DISP=(NEW,CATLG,CATLG),'
    PUSH LINE
    LINE='//SORTWKØ4 DD DSN=SYSPDBA.PSØ.RECI'||JOB_CNT||,
         '.WORK4.TEMP,'
    PUSH LINE
    LINE='//            SPACE=(1Ø24,('||SPACE_FOR_IX_REC1||,
         ','||SPACE_FOR_IX_REC1||'),,,ROUND)'
    PUSH LINE
    LINE='//            DISP=(NEW,CATLG,CATLG),'
    PUSH LINE
    LINE='//SORTWKØ3 DD DSN=SYSPDBA.PSØ.RECI'||JOB_CNT||,
         '.WORK3.TEMP,'
    PUSH LINE
  END
  LINE='//            SPACE=(1Ø24,('||SPACE_FOR_IX_REC1||,
       ','||SPACE_FOR_IX_REC1||'),,,ROUND)'
  PUSH LINE
  IF SPACE_FOR_IX_REC1 > MAX_KB_WORK_CATLG THEN DO
    LINE='//            DISP=(NEW,CATLG,CATLG),'
    PUSH LINE
    LINE='//SORTWKØ2 DD DSN=SYSPDBA.PSØ.RECI'||JOB_CNT||,
         '.WORK2.TEMP,'
    PUSH LINE
  END
  ELSE DO
    LINE='//            DISP=(NEW,DELETE,CATLG),'
    PUSH LINE
    LINE='//SORTWKØ2 DD DSN=SYSPDBA.PSØ.RECI'||JOB_CNT||,
         '.WORK2.TEMP,'
    PUSH LINE
  END
  LINE='//            SPACE=(1Ø24,('||SPACE_FOR_IX_REC1||,
       ','||SPACE_FOR_IX_REC1||'),,,ROUND)'
  PUSH LINE
  IF SPACE_FOR_IX_REC1 > MAX_KB_WORK_CATLG THEN DO
    LINE='//            DISP=(NEW,CATLG,CATLG),'
    PUSH LINE
    LINE='//SORTWKØ1 DD DSN=SYSPDBA.PSØ.RECI'||JOB_CNT||,
         '.WORK1.TEMP,'
    PUSH LINE
  END
  ELSE DO
    LINE='//            DISP=(NEW,DELETE,CATLG),'
    PUSH LINE
    LINE='//SORTWKØ1 DD DSN=SYSPDBA.PSØ.RECI'||JOB_CNT||,
         '.WORK1.TEMP,'
    PUSH LINE
  END
  LINE='//STEPLIB  DD DSN='||DB2ID||'DSN.SDSNLOAD,DISP=SHR'
  PUSH LINE
  LINE='//RECIX'||STEP_CNT||'  EXEC PGM=DSNUTILB,REGION=ØM,'||,
       'PARM='||''''||'DB'||DB2ID||'Ø'||',RECIX'||JOB_CNT''''
  PUSH LINE
  £EXECIO * DISKW REIXJCL£
RETURN
JOB_CARD_RECOVER_IX:PROCEDURE EXPOSE DB2ID JOB_CNT TARIH SAAT
  PUSH ''
  PUSH '//*       CREATED ON '||TARIH||'   '||SAAT
  PUSH '// MSGLEVEL=(1,1),REGION=ØM,NOTIFY=SKMXSYP,TYPRUN=HOLD'
  LINE='//RECJIX'||  JOB_CNT ||,
        '  JOB (ACCT#),'RECIX',MSGCLASS=X,CLASS=9,'
  PUSH LINE
  £EXECIO * DISKW REIXJCL£
RETURN
DO_SELECT:PROCEDURE EXPOSE SEL. DB2ID
  PUSH ''
  DO X = SEL.Ø TO 1 BY -1
    PUSH SEL.X
  END
  £EXECIO * DISKW SYSIN (FINIS£
  CMD = £RUN PROGRAM(DSNTIAUL) PLAN(DSNTIAUL)£
  CMD = CMD || ' LIB('''||DB2ID||'DSN.RUNLIB.LOAD'')'
  CMD = CMD || £ PARMS('SQL')£
  QUEUE 'END '
  IF DB2ID='D' THEN 'DSN SYSTEM(DBDØ)'
  IF DB2ID='T' THEN 'DSN SYSTEM(DBTØ)'
  IF DB2ID='E' THEN 'DSN SYSTEM(DBEØ)'
  IF DB2ID='G' THEN 'DSN SYSTEM(DBGØ)'
  IF DB2ID='P' THEN 'DSN SYSTEM(DBPØ)'
  IF RC > Ø THEN DO
    SAY 'CAN NOT CONNECT TO DB2 SUBSYSTEM.'
    SAY 'PLEASE TRY LATER...'
    RETURN
  END
  QUEUE CMD
  QUEUE 'END '
  IF DB2ID='D' THEN 'DSN SYSTEM(DBDØ)'
  IF DB2ID='T' THEN 'DSN SYSTEM(DBTØ)'
  IF DB2ID='E' THEN 'DSN SYSTEM(DBEØ)'
  IF DB2ID='G' THEN 'DSN SYSTEM(DBGØ)'
  IF DB2ID='P' THEN 'DSN SYSTEM(DBPØ)'
  £EXECIO * DISKR SYSPRINT (STEM SQLHATA.£
  UNLD_OK=Ø
  DO SQ_LP=1 TO SQLHATA.Ø
    IF INDEX(SQLHATA.SQ_LP,'DSNT495I SUCCESSFUL UNLOAD') > Ø THEN ,
      UNLD_OK=1
  END
  IF UNLD_OK=Ø THEN DO
    SAY 'THERE IS AN ERROR IN SQL STATEMENT.'
    DO SQ_LP2=1 TO SQLHATA.Ø
      SAY SQLHATA.SQ_LP2
    END
    EXIT 2Ø
  END
  PUSH ''
  £EXECIO * DISKW SYSPRINT (FINIS£
RETURN
ADD_WTO:PROCEDURE
  PUSH ''
  PUSH '/*'
  PUSH '->*************************************************************'
  PUSH '-> DO NOT SUBMIT IT AT THE LOCAL SITE.'
  PUSH '-> THIS JCL IS PREPARED FOR DISASTER RECOVERY PURPOSES.'
  PUSH '->*************************************************************'
  PUSH '//SYSIN    DD   *'
  PUSH '//ERROR    EXEC IPOWTO,REGION=ØM,COND=((4,LE),EVEN)'
  £EXECIO * DISKW RETSJCL£
RETURN
WRITE_SYSIN:
  PUSH ''
  PUSH '//DSNUPROC.SYSIN DD *'
  £EXECIO * DISKW RETSJCL£
  DO LP1=FIRST_CNT TO LAST_CNT
    IF NUMPART.LP1 = Ø THEN DO
      PUSH ''
      IF OP_ID='TOCOPY' THEN DO
        LINE = '   TOCOPY '||LAST_COPY_DSN.LP1
        PUSH LINE
      END
      LINE = '   RECOVER TABLESPACE ' || DBNAME.LP1||,
             '.'||TSNAME.LP1
      PUSH LINE
      £EXECIO * DISKW RETSJCL£
    END;
    ELSE DO;
      PUSH ''
      IF OP_ID='TOCOPY' THEN ,
        LINE = '   DSNUM '||NUMPART.LP1||' TOCOPY '||,
                 LAST_COPY_DSN.LP1
      ELSE LINE = '   DSNUM '||NUMPART.LP1
      PUSH LINE
      LINE = '   RECOVER TABLESPACE ' || DBNAME.LP1||,
             '.'||TSNAME.LP1
      PUSH LINE
      £EXECIO * DISKW RETSJCL£
    END;
  END
  PUSH ''
  PUSH '/*'
  £EXECIO * DISKW RETSJCL£
RETURN;
```

RECALLX

```
//RECALLX JOB (ACCT#),'',MSGCLASS=X,CLASS=P,
// MSGLEVEL=(1,1),REGION=4M
//*****************************************************************
//* RECALL   DB2ID OP_ID REC_DSNAME TSSELECT
//* DB2ID  : DB2 SUBSYSTEM ID. IT MAY BE D, T, E  OR P
//* OP_ID  : TOCOPY OR ENDOFLOG
//* RECTS_JCL_NAME  : RECOVER TABLESPACE JCL TO BE CREATED.
//* RECIX_JCL_NAME  : RECOVER INDEX JCL TO BE CREATED.
//* JOBCNT          : HOW MANY SEPARATE JOBS WILL RECOVER INDEXES.
//* PARTITION       : 'YES' IF RECOVERY IS MADE PARTITIONED
//*                    TABLESPACE LEVEL, OTHERWISE 'NO'.
//*****************************************************************
//RUNEXEC  EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø,REGION=8192K
//STEPLIB  DD DSN=ISP.SISPLOAD,DISP=SHR
//         DD DSN=PDSN.SDSNLOAD,DISP=SHR
//SYSEXEC  DD   DSN=SYSPDBA.REXXLIB,DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
 EXECUTIL SEARCHDD(YES)
 %RECALL P TOCOPY RECTSUPD RECIXUPD 4 NO
/*
```

Abdullah Ongul
DBA
Disbank (Turkey)

Many subscribers reading *DB2 Update* will have met similar problems and come up with quite different solutions. We'd like to hear what your alternative solution is. Contact the editor, Trevor Eddolls, at any of the addresses shown on page 2 for a copy of our *Notes for Contributors*.

DB2 news

Embarcadero Europe has announced the availability of its ERJStudio 4.0, an upgrade to its modelling environment that offers advanced parser-based support for stored procedures and triggers and an automation interface for user customization.

Users can create stored procedures and triggers in their native DBMS languages (including PL/SQL and Transact SQL) or they can create template versions for drag-and-drop reuse. ER/Studio 4.0 ensures consistent stored procedure and trigger object dependency for tables, views, and other procedural logic.

ER/Studio has an automation interface and, by creating Sax BASIC commands (a VBA-like language) in ER/Studio's new macro scripting UI, users can tap directly into ER/Studio's own object model to expand existing functionality and create new functionality.

The automation interface can be used to create operations internal to ER/Studio, such as automating repetitive tasks, or between ER/Studio and virtually any other database or application environment with an exposed API or similar automation interface.

ER/Studio 4.0 runs on Windows 95, 98, NT, and 2000. Notational and method support includes IDEF1X, James Martin's IE, and Filtered IE (designed to hide foreign keys). Supported databases include DB2 and DB2 Universal Database, as well as Oracle 7.3, 8, and 8*i*, Sybase 1 l.*x*, Informix SE and Online, Microsoft SQL Server 6.5 and 7.0, SQL Anywhere 5, Watcom 4, InterBase 4.*x*, Access 2.0, 95, 97, and 2000, and Visual FoxPro.

For further information contact:
Embarcadero, 400 Montgomery St #300, San Francisco, CA 94104, USA.
Tel: (415) 834 3131.
URL: http://www.embarcadero.com.

* * *

IBM has announced DB2 UDB Version 7, which it describes as being "designed from the ground up for dot coms". The new version has an integrated in-memory text search engine, which is said to offer a ten-fold performance increase over existing techniques.

Version 7 has 'deep' XML integration facilities with intelligent searching and automated management with an integrated DB2 datatype. There is claimed easier access to heterogeneous data sources with an integrated distributed query capability and tools such as DB2 Relational Connect and an enhanced DB2 Data Links Manager.

The new database is said to be Windows 2000-ready with expanded OLE DB support and integration with Visual Studio development tools. It also comes with a Java Transaction API and it supports JDBC V2.

In terms of BI function, Version 7 has an integrated data warehouse centre with a claimed easy-to-use GUI and launchpad, heterogeneous source access, and industry-standard metadata management. There's a new OLAP Starter Kit, based on Hyperion Essbase technology, and new query capabilities with relational OLAP functions in SQL.

For further information contact your local IBM representative.