81

DB2

July 1999

In this issue

update

DB2 Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
E-mail: info@xephon.com

North American office

Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75077-2150
USA
Telephone: 940 455 7050

Contributions

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

DB2 Update on-line

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

Editor

Robert Burgess

Disclaimer

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

Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs £255.00 in the UK; $380.00 in the USA and Canada; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1994 issue, are available separately to subscribers for £22.50 ($33.50) each including postage.

Converting Type 1 indexes to Type 2

DB2 Version 6 will not support Type 1 indexes, so all Type 1 indexes must be converted to Type 2 before migrating to Version 6. IBM has provided the CATMAINT utility, which converts catalog and directory indexes from Type 1 to Type 2; however, it only converts IBM-defined catalog indexes.

I have written the following REXX procedure for user-defined indexes. The steps are:

1 ALTER INDEX index-name CONVERT TO TYPE 2.

  The index is left in recover pending state (SQLCODE=610), and the index change does not take place until the index is rebuilt by a RECOVER utility.

2 RECOVER INDEX(index-name).

TI2 PROCEDURE

Figure 1 shows the TI2 Parameter Entry panel. To the right of the panel is the 'Prompt' column, which should help to explain which values to enter in the Parameter Value column. 'Service' is the service you want to execute, and has the following valid parameters:

- IREC – Recovery index(es).
- IREO – Reorg index(es).
- IRUN – Runstats index(es).
- CONV – convert Type 1 to Type 2 index.

'Creator', 'Name', 'Dbname', 'Tbcreator', and 'Tbname' are DB2 Catalog search conditions.

COMPONENTS OF TI2

TI2 consists of the following components:

- TI2 is the driver procedure.

```
DB2 INDEX   Recovery, Reorg, Runstat, and Convert Type 1 to Type 2

Command ===>

PARAMETER   PARAMETER VALUE                PROMPT

SSID     => DSNN                           DB2 Sub-System Identifier
Service  => CONV                           IREC, IREO, IRUN or CONV
Creator  => _________                      Index Creator
Name     => ___________________            Index Name
Dbname   => _________                      Database Name
Tbcreator=> BZVER                          Table Creator
Tbname   => TLØ57                          Table Name

            Enter parameter values for the Index service!

                                PF3 Return
```

Figure 1: Parameter Entry panel

- TYPEI2P is the main menu.
- TYPEI2L is the selection result.
- TYPEI00 is the TI2 message.
- PTYPEI2 is the PL/I source code.
- DBINDEX is the JCL skeleton.

TI2

```
/* REXX */
/* DB2 INDEX: Recovery, Reorg, Runstat, and Convert Type 1 to Type 2 */
/* trace r */
 zpfctl = 'OFF'
 Y=MSG("OFF")
 /*****************************************************/
 /* Change to your convention standards               */
 program = 'PTYPEI2'
 plan    = 'PTYPEI2'
 llib    = 'your.LOADLIB'
 /*****************************************************/
 address ispexec 'vput (zpfctl) profile'
 Call Aloc
 cur='icrec'
```

```
TOP:
address ispexec "display panel(typei2p) cursor("CUR")"
if rc=8 then do
   Call Free_proc
   exit
end
/* Check input parameters                              */
if serv='IREC' | serv='IREO' | serv='IRUN' | serv='CONV' then nop
else do
   message='Invalid service parameter.',
   'Valid values are: IREC-recovery, IREO-reorg, IRUN-runstat',
   'or CONV-convert Type 1 to Type 2 index'.
   Call Error 'serv'
end
if icrec=' ' & iname=' ' & dbnc=' ' & crec=' ' & tabc=' ' then do
   message='At least one Catalog search field must be entered.'
   Call Error 'icrec'
end
parm=substr(icrec,1,8)||substr(iname,1,18)||substr(dbnc,1,8)||,
     substr(crec,1,8)||substr(tabc,1,18)||serv
ADDRESS TSO
QUEUE "RUN PROGRAM("program") PLAN("plan"),
       LIBRARY ('"llib"'),
       PARMS ('/"parm"')"
QUEUE "END "
"DSN SYSTEM("db2")"
if rc=12 then do
   "delstack"
   Call Free_proc
   Call Aloc
   message = 'Error.   'db2||' ssid is not valid  |'
   Call Error 'db2'
END
"EXECIO * DISKR SYSPRINT (STEM ROW."
if substr(row.1,2) = 'NO CATALOG ENTRIES FOUND' then do
   Call Free_proc
   Call Aloc
   message = 'No catalog entries found, check Search Fields.',
             'Perhaps Type 1 index(es) do not exist.'
   Call Error 'dbnc'
end
else do
   address ispexec 'tbcreate "ilist",
           names(v1 v2 v3 v4 v5 vcat dbname space pr pri sec)'
   num=row.Ø
   do i=1 to row.Ø
      v1     = word(row.i,2)
      v2     = word(row.i,3)
      v3     = word(row.i,4)
      v4     = word(row.i,5)
      v5     = right(word(row.i,6),16)
```

```
      if serv='IREO' then Call Calculate
      address ispexec 'tbadd "ilist"'
   end
   address ispexec 'tbtop "ilist"'
   address ispexec 'addpop row(1) column(5)'
   address ispexec 'tbdispl "ilist" panel(typei2l)'
   if rc=8 then do
      Call Free_proc
      address ispexec 'tbend "ilist"'
      Call Aloc
      address ispexec rempop all
      signal top
   end
end
address ispexec rempop all
Call Free_proc
/* JCL Skeleton DB2 Index    */
if serv='IREC' then title = 'RECOVERY INDEX UTILITY'
if serv='IREO' then title = 'REORG INDEX UTILITY'
if serv='IRUN' then title = 'RUNSTAT INDEX UTILITY'
if serv='CONV' then title = 'CONVERT INDEX UTILITY'
date=date()
time=time(c)
user=userid()
suf='D'||right(date('D'),3,'0')||right(time('M'),4,'0')
tempfile=userid()||'.UTIL.INDEX.TEMP'
address tso
"delete '"tempfile"'"
"free dsname('"tempfile"')"
"free ddname(ispfile)"
"free attrlist(formfile)"
"attrib formfile blksize(800) lrecl(80) recfm(f b) dsorg(ps)"
"alloc ddname(ispfile) dsname('"tempfile"')",
       "new using (formfile) unit(3390) space(1 1) cylinders"
address ispexec
"ftopen"
"ftincl DBINDEX"
"ftclose"
zedsmsg = "JCL shown"
zedlmsg = "JCL DB2 Index shown"
"setmsg msg(isrz001)"
"edit dataset('"tempfile"')"
address ispexec 'tbend "ilist"'
exit
Aloc:
  ADDRESS TSO "DELETE '"SYSVAR(SYSUID)".UTIL.INDEX'"
  "ALLOC DD(SYSPRINT) DSN('"SYSVAR(SYSUID)".UTIL.INDEX') SPACE(24 8),
  TRACK MOD UNIT(3390) RECFM(F,B) LRECL(120) BLKSIZE(1200) ,
  F(SYSPRINT) CATALOG REUSE "
Return
Error:
```

```
   ARG cur_par
   cur=cur_par
   address ispexec "setmsg msg(typei001)"
   signal top
 Return
 Free_proc:
   "execio 0 diskr sysprint (finis"
   address tso "free f(sysprint)"
 Return
 Calculate:
   pri=0
   sec=0
   vcat   = word(row.i,7)
   dbname = word(row.i,8)
   space  = word(row.i,9)
   pr     = word(row.i,10)
   if pr  = 0 then pr=1
   do j=1 to pr
      part='.I0001.A'||right(j,3,'0')
      file=vcat||'.DSNDBD.'||strip(dbname)||'.'||strip(space)||part
      dsn = "('"file"')"
      X=OUTTRAP('var.')
      address tso "listc" entries dsn allocation
      X=OUTTRAP('OFF')
      if rc=0 then do
         hurba = word(translate(var.9,' ','-'),7)
         if hurba < trunc(737280/15,0) then do
            pri=pri+1
            sec=sec+1
         end
         else do
            pri=pri+trunc((hurba/(737280/15)+1),0)
            sec=sec+max(trunc(pri*0.2,0),1)
         end
      end
   end
 Return
```

TYPEI2P

```
)Attr Default(%+_)
   | type(text)   intens(high) caps(on ) color(yellow)
   $ type(output) intens(high) caps(off) color(yellow)
   ? type(text)   intens(high) caps(on ) color(green) hilite(reverse)
   # type(text)   intens(high) caps(off) hilite(reverse)
   } type(text)   intens(high) caps(off) color(yellow) hilite(reverse)
   [ type( input) intens(high) caps(on ) color(green) pad(_)
)Body  Expand(//)
? DB2 INDEX } Recovery, Reorg, Runstat, and Convert Type 1 to Type 2 +
+
%Command ===>_zcmd                                                    +
```

```
+
#PARAMETER  #PARAMETER VALUE                  #PROMPT                    +
+
+SSID     =>[db2 +                           DB2 Sub-System Identifier
+Service  =>[serv+                           IREC, IREO, IRUN, or CONV
+Creator  =>[icrec   +                       Index Creator
+Name     =>[iname           +               Index Name
+Dbname   =>[dbnc    +                       Database Name
+Tbcreator=>[crec    +                       Table Creator
+Tbname   =>[tabc            +               Table Name
+
            $msg                                                        +
+
                                } PF3 Return +
)Init
  if (&db2 ¬= ' ')
      .attr (db2) = 'pad(nulls)'
  if (&serv ¬= ' ')
      .attr (serv) = 'pad(nulls)'
  if (&icrec ¬= ' ')
      .attr (icrec) = 'pad(nulls)'
  if (&iname ¬= ' ')
      .attr (iname) = 'pad(nulls)'
  if (&dbnc ¬= ' ')
      .attr (dbnc) = 'pad(nulls)'
  if (&crec ¬= ' ')
      .attr (crec) = 'pad(nulls)'
  if (&tabc ¬= ' ')
      .attr (tabc) = 'pad(nulls)'
  &msg = 'Enter parameter values for the Index service!'
)Reinit
)Proc
  VPUT (db2 serv icrec iname dbnc crec tabc ) PROFILE
)End
```

TYPEI2L

```
)Attr Default(%+_)
   ( type(text  ) intens(high) hilite(reverse)
   ] type(text  ) intens(high) hilite(reverse) color(green)
   / type(text  ) intens(high) hilite(reverse) color(yellow)
   ~ type(output) intens(high)                 color(red)
   [ type(text  ) intens(high) hilite(reverse) color(white) caps(off)
   + type(text  ) intens(low )
   _ type( input) intens(high) caps(on ) just(left )
   ¬ type(output) intens(low ) caps(off) just(asis )
)Body window(74,19)
[Selection Result+
+
+Command ===>_zcmd                                  +Scroll ===>_amt +
+
```

```
+Press]Enter+to have this service continue.
+Press]End  +to respecify your PARAMETERS.
+
]Index             ]Creator ]Tbname          ]Creator ]            Card+
+
)Model
¬z                 ¬z       ¬z               ¬z       ¬z              +
)Init
  .ZVARS = '(v1 v2 v3 v4 v5)'
  &amt = PAGE
)Reinit
)Proc
)End
```

TYPEI00

```
TYPEIØØ1          .ALARM = YES  .WINDOW=NORESP .ALARM = YES
'&message'
```

PTYPEI2

```
* PROCESS GS,OFFSET,OPT(TIME);
 PTYPEI2:PROC(PARMS)OPTIONS(MAIN) REORDER;
/*****************************************************************/
/* DESCRIPTION: PL/I PROGRAM - DB2 INDEX SELECTION RESULT        */
/*****************************************************************/
  DCL PARMS CHAR(1ØØ) VAR;
  DCL SYSPRINT    FILE STREAM OUTPUT;
  DCL NUMSEQ        BIN FIXED(31) INIT(Ø);
  DCL MCARD         PIC'—.—.—.—9';
  DCL MPART         PIC'—9';
/*****************************************************************/
/* DCLGEN TABLE: SYSIBM.SYSINDEXES                               */
/*****************************************************************/
  DCL 1 DCLW,
      5 NAME             CHAR(18) VAR,
      5 CREATOR          CHAR(8),
      5 TBNAME           CHAR(18) VAR,
      5 TBCREATOR        CHAR(8),
      5 VCAT             CHAR(8),
      5 DBNAME           CHAR(8),
      5 ISPACE           CHAR(8),
      5 ITYPE            CHAR(1),
      5 PART             BIN FIXED(15),
      5 CARD             BIN FIXED(31);
  DCL 1 WORKST,
      2 ICREC            CHAR(8)  VAR,
      2 INAME            CHAR(18) VAR,
      2 DBNC             CHAR(8)  VAR,
      2 CREC             CHAR(8)  VAR,
```

```
    2 TABC             CHAR(18) VAR,
    2 SERV             CHAR(4);

DCL (SUBSTR,DATE,TIME,NULL,ADDR,LENGTH,INDEX) BUILTIN;
DCL IC             BIN FIXED(15);
DCL OUT            CHAR(18) VAR;

EXEC SQL INCLUDE SQLCA;
IF SUBSTR(PARMS,1,8)=' ' THEN ICREC='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,1,8),OUT);
   ICREC=OUT;
   IF LENGTH(ICREC) < 8 THEN ICREC=ICREC||'%';
END;
IF SUBSTR(PARMS,9,18)=' ' THEN INAME='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,9,18),OUT);
   INAME=OUT;
   IF LENGTH(INAME) < 18 THEN INAME=INAME||'%';
END;
IF SUBSTR(PARMS,27,8)=' ' THEN DBNC='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,27,8),OUT);
   DBNC=OUT;
   IF LENGTH(DBNC) < 8 THEN DBNC=DBNC||'%';
END;
IF SUBSTR(PARMS,35,8)=' ' THEN CREC='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,35,8),OUT);
   CREC=OUT;
   IF LENGTH(CREC) < 8 THEN CREC=CREC||'%';
END;
IF SUBSTR(PARMS,43,18)=' ' THEN TABC='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,43,8),OUT);
   TABC=OUT;
   IF LENGTH(TABC) < 8 THEN TABC=TABC||'%';
END;
SERV = SUBSTR(PARMS,61,4);
ITYPE='%';
IF SERV='CONV' THEN ITYPE=' ';

/* SELECTION RESULTS                                  */
EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR SELECT
  NAME, CREATOR, TBNAME, TBCREATOR, VCATNAME,
  DBNAME, INDEXSPACE, SUM(CARD), MAX(PARTITION)
FROM SYSIBM.SYSINDEXES I,
     SYSIBM.SYSINDEXPART
WHERE CREATOR   LIKE :ICREC
  AND NAME      LIKE :INAME
```

```
    AND DBNAME    LIKE :DBNC
    AND TBCREATOR LIKE :CREC
    AND TBNAME    LIKE :TABC
    AND I.INDEXTYPE LIKE :ITYPE
    AND CREATOR=IXCREATOR
    AND NAME   =IXNAME
  GROUP BY NAME, CREATOR, TBNAME, TBCREATOR,
           VCATNAME, DBNAME, INDEXSPACE
  ORDER BY TBCREATOR, TBNAME, NAME
  FOR FETCH ONLY;
  EXEC SQL OPEN C1;

  CALL FETCH;
  DO WHILE (SQLCODE=Ø);
     NUMSEQ=1;
     MCARD=CARD;
     MPART=PART;
     PUT SKIP LIST ('I '||NAME||' '||CREATOR||' '||TBNAME||' '||
         TBCREATOR||' '||MCARD||' '||VCAT||' '||DBNAME||' '||
         ISPACE||' '||MPART);
     CALL FETCH;
  END;
  EXEC SQL CLOSE C1;
  IF NUMSEQ=Ø THEN PUT SKIP LIST ('NO CATALOG ENTRIES FOUND');

  FETCH:PROC;
     EXEC SQL FETCH C1 INTO
          :NAME, :CREATOR, :TBNAME, :TBCREATOR,
          :VCAT, :DBNAME, :ISPACE, :CARD, :PART;
  END FETCH;
  FUNC:PROC(INP,OUT);
       DCL INP CHAR(18);
       DCL OUT CHAR(18) VAR;
       DO IC=1 TO 18 BY 1 WHILE (SUBSTR(INP,IC,1) ¬=' ');
       END;
       OUT=SUBSTR(INP,1,IC-1);
   END FUNC;
 END PTYPEI2;
```

DBINDEX

```
)TBA 72
)CM ------------------------------------------------------------------------
)CM Skeleton to generate JCL for DB2 Index                                 -
)CM ------------------------------------------------------------------------
//&user.X JOB (ACCT#),'&option',
//             NOTIFY=&user,REGION=4M,
//             CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//* ******************************************************************
```

```
//*    &title
//*    GENERATION DATE AND TIME : &date AT: &time
//*
//*    INDEX UTILITY - WAS RUN WITH THE FOLLOWING PARAMETERS:
//*    PARAMETER   PARAMETER VALUE
//*    _____       ____________
//*    SSID     :  &db2
//*    Service  :  &serv
//*    Creator  :  &icrec
//*    Name     :  &iname
//*    Dbname   :  &dbnc
//*    Tbcreator:  &crec
//*    Tbname   :  &tabc
//* ********************************************************************
)SEL &serv EQ CONV
//RUNSQL  EXEC PGM=IKJEFT01
//STEPLIB DD DISP=SHR,DSN=DSN510.SDSNLOAD
//        DD DISP=SHR,DSN=CEE.SCEERUN
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(&db2)
 RUN PROGRAM(DSNTEP2) PLAN(DSNTEP51) -
     LIB('DSN510.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *
  SET CURRENT SQLID = '&user' ;
)DOT "ILIST"
  ALTER INDEX &v2..&v1 CONVERT TO TYPE 2 ;
)ENDDOT
  COMMIT;
)ENDSEL
)SEL &serv EQ IREC OR &serv EQ CONV
//*-- RECOVER INDEX ------------------
//IRECOV EXEC DSNUPROC,SYSTEM=&db2,
//      UID='&user..IRECOV',UTPROC=''
//STEPLIB  DD DSN=DSN510.SDSNLOAD,DISP=SHR
//SORTWK01 DD DSN=&SORTWK1,
//     DISP=(NEW,DELETE,DELETE),SPACE=(CYL,(30,30),,,ROUND),
//     UNIT=3390
//SORTWK02 DD DSN=&SORTWK2,
//     DISP=(NEW,DELETE,DELETE),SPACE=(CYL,(30,30),,,ROUND),
//     UNIT=3390
//SORTWK03 DD DSN=&SORTWK3,
//     DISP=(NEW,DELETE,DELETE),SPACE=(CYL,(30,30),,,ROUND),
//     UNIT=3390
//SORTWK04 DD DSN=&SORTWK4,
//     DISP=(NEW,DELETE,DELETE),SPACE=(CYL,(30,30),,,ROUND),
//     UNIT=3390
//SORTOUT DD DSN=&SORTOUT,
```

```
//     DISP=(NEW,DELETE,DELETE),SPACE=(CYL,(30,30),,,ROUND),
//     UNIT=3390
//SYSUT1 DD DSN=&SYSUT1,
//     DISP=(NEW,DELETE,DELETE),SPACE=(CYL,(30,30),,,ROUND),
//     UNIT=3390
//SYSIN    DD  *
)DOT "ILIST"
RECOVER INDEX(&v2..&v1)
)ENDDOT
)ENDSEL
)SEL &serv EQ IREO
//*-- REORG INDEDX ------------------
)SET inc = 0
)DOT "ILIST"
)SET inc = &inc + 1
//IREO&inc   EXEC DSNUPROC,SYSTEM=&db2,
//      UID='&user..IREO&inc',UTPROC=''
//STEPLIB  DD DSN=DSN510.SDSNLOAD,DISP=SHR
//SORTOUT DD DISP=(NEW,DELETE,CATLG),
//        UNIT=3390,
//        DSN=&user..IRSOUT.&suf,
//        SPACE=(TRK,(&pri,&sec),,,ROUND)
//SYSUT1  DD DISP=(NEW,DELETE,CATLG),
//        UNIT=3390,
//        DSN=&user..IRSUT1.&suf,
//        SPACE=(TRK,(&pri,&sec),,,ROUND)
//SYSIN    DD  *
REORG    INDEX  &v2..&v1
                SORTDEVT 3390
                SORTNUM  5
RUNSTATS INDEX (&v2..&v1)
)ENDDOT
)ENDSEL
)SEL &serv EQ IRUN
//*-- RUNSTATS INDEX ----------------
//IRUNST EXEC DSNUPROC,SYSTEM=&db2,
//      UID='&user..IRUNST',UTPROC=''
//STEPLIB  DD DSN=DSN510.SDSNLOAD,DISP=SHR
//SYSIN    DD  *
)DOT "ILIST"
RUNSTATS INDEX (&v2..&v1)
               REPORT NO
               UPDATE ALL
)ENDDOT
)ENDSEL
/*
```

Bernard Zver
Database Administrator
Informatika Maribor (Slovenia)

Data warehousing guidelines for DB2

More and more organizations are building their data warehouses using DB2 for OS/390 because of the scalability, reliability, and robust architecture that it provides. You can use the following guidelines as rules of thumb when you're designing, implementing, and using your DB2-based data warehouse. Although some of the advice is platform-independent and useful regardless of the DBMS being used to build your data warehouse, the guidelines were written with DB2 for OS/390 specifically in mind.

DO NOT IMPLEMENT A DATA WAREHOUSE AS A PANACEA

Many data warehouse development projects begin with 'pie in the sky' expectations. One of the biggest problems with a data warehouse project is the situation in which the data warehouse is viewed as a 'magic bullet' that will solve all of management's information problems.

To alleviate this type of problem, you should manage expectations by securing an executive sponsor, limiting the scope of the project, and implementing the data warehouse in stages (or possibly by implementing multiple data marts for each department).

DO NOT BECOME 100% TECHNOLOGY-FOCUSED

When you're developing a data warehouse, be sure to include tools, people, and methods in your warehouse blueprint. Too often, the focus is solely on the technology and tools aspect. To be successful, a data warehouse project requires more than just sound technology. You need careful planning and implementation (methods) as well as a means to learn from the efforts of others (people) through mentoring, consulting, education, seminars, and user groups.

DO NOT MIX OPERATIONAL NEEDS INTO THE PROJECT

When a data warehousing project is first initiated, it may have a mixture of operational and analytical/informational objectives. This mixture is a recipe for disaster. Redefine the project to concentrate on

non-operational, informational needs only. The primary reason for the existence of the data warehouse in the first place is to segregate operational processing from reporting.

ENSURE READ-ONLY DATA

Create the data warehouse as a decision support vehicle. The data should be periodically updated and summarized. If your design calls for a data warehouse in which all the data is modified immediately as it is changed in production, you need to rethink your data warehouse design.

Consider starting DB2 data warehouse databases as 'ACCESS(RO)' to ensure read-only access. Doing so has the additional effect of eliminating locking on the read-only databases. When the data warehouse is refreshed, the databases have to be restarted in read/write mode.

CONSIDER USING DIRTY READS

Because data warehouses are read-only in nature, locking is not truly required. You can specify 'ISOLATION(UR)' for all plans, packages, and queries used in the data warehouse environment. With 'ISOLATION(UR)', DB2 will take fewer locks, thereby enhancing performance. However, DB2 may read uncommitted data when 'ISOLATION(UR)' is specified. This should not be a major concern in the read-only data warehouse.

BE AWARE OF THE COMPLEXITY OF IMPLEMENTATION

Moving data into a data warehouse is a complex task. Detailed knowledge of the applications accessing the source databases that feed the data warehouse must be available. Be sure to allot development time for learning the complexities of the source systems. Frequently, the systems documentation for a production system is inadequate or non-existent.

Additionally, be sure to analyse the source data to determine what level of data scrubbing is required. This process can be an immense, time-consuming task.

PREPARE TO MANAGE DATA QUALITY ISSUES CONSTANTLY

Maintaining data quality will be an on-going concern. Both the end users and the data warehouse construction and maintenance team are responsible for promoting and fostering data quality. Data problems will be discovered not only throughout the development phase of the data warehouse, but throughout the useful life of the data warehouse.

Be sure to establish a policy for reporting and correcting data anomalies before the data warehouse is made generally available to its end users. Additionally, be sure to involve the end users in the creation and support of this policy; otherwise, it is doomed to fail. The end users understand the data better than anyone else in the organization, including the data warehouse developers and DBAs.

DO NOT OPERATE IN A VACUUM

As business needs change, operational systems change. When operational data stores change, the data warehouse will be affected as well. When a data warehouse is involved, however, both the operational database and the data warehouse must be analysed for the impact of changing any data formats. This is true because the data warehouse stores historical data that you might not be able to change to the new format. Before the change is made to the operational system, the data warehouse team must be prepared firstly to accept the new format as input to the data warehouse, and secondly, to either maintain multiple data formats for the changed data element or to implement a conversion mechanism as part of the data transformation process. Conversion, however, can result in lost or confusing data.

TACKLE OPERATIONAL PROBLEMS IN THE PROJECT

You will encounter problems in operational systems that feed the data warehouse. These problems may have been in production for a year, running undetected. The data warehousing project will uncover many such errors. Be prepared to find them and have a plan for handling them.

Only three options are available:

- Ignore the problem with the understanding that the problem will exist in the data warehouse if not corrected.

- Fix the problem in the operational system.
- If possible, fix the problem during the data transformation phase of data warehouse population.

Of course, the second and third options are the favoured approaches.

DETERMINE WHEN DATA IS TO BE PURGED

Even in the data warehouse environment, when certain thresholds are reached, maintaining certain data in the data warehouse does not make sense. This situation may occur because of technology reasons (such as reaching a capacity limit), regulatory reasons (change in regulations or laws), or business reasons (restructuring data, instituting different processes, and so on).

Plan to arrange for methods of purging data from the data warehouse without dropping the data forever. A good tactic is to prepare a generic plan for offloading warehouse data to tape or optical disk.

USE DENORMALIZATION STRATEGIES

Experiment with denormalized tables. The opposite of normalization, denormalization is the process of putting one fact in many places. Because the data warehouse is a read-only database, you should optimize query at the expense of update. Denormalization will achieve this. Analyse the data access requirements of the most frequent queries, and plan to denormalize to optimize those queries.

There are ten types of denormalization that can be useful when implementing DB2-based data warehouses:

- Prejoined tables – combining two tables together into a single table when the cost of joining is prohibitive.
- Report tables – creating a table to store specialized critical reports that require fast access.
- Mirror tables – creating copies of tables when the data is required concurrently by two types of environment.
- Split tables – breaking a table into two parts when distinct groups use different parts of the table.

- Combined tables – combining two tables together when one-to-one relationships exist.
- Redundant data – carrying redundant columns in multiple tables to reduce the number of table joins required.
- Repeating groups – storing repeating groups in a single row to reduce I/O and (possibly) DASD usage.
- Derivable data – storing calculated results to eliminate calculations and algorithms.
- To avoid BP32K – splitting columns of very large rows across multiple tables to avoid using pages larger than 4KB in size.
- Speed tables – storing traversed hierarchies to support bill-of-material processing.

When you design the data warehouses you should be alert for situations where each of these types of denormalization may be useful. In general, denormalization speeds data retrieval, which is desirable for a data warehouse. However, denormalize only when a completely normalized design will not perform optimally.

BE GENEROUS WITH INDEXES

The use of indexes is a major factor in creating efficient data retrieval. You can usually use indexes more liberally in the read-only setting of the data warehouse. Remember, though, you must make a trade-off between data loading and modification and the number of indexes.

The data warehouse indexes do not have to be the same indexes that exist in the operational system, even if the data warehouse is nothing more than an exact replica or snapshot of the operational databases. You should optimize the indexes based on the access patterns and query needs of the decision support environment of the data warehouse. Also, use type 2 indexes to remove index locking as a consideration for the data warehouse.

AVOID REFERENTIAL INTEGRITY AND CHECK CONSTRAINTS

Because data is cleansed and scrubbed during the data transformation process, implementing data integrity mechanisms such as referential

integrity (RI) and check constraints on data warehouse tables is not efficient. Even without a comprehensive cleansing during data transformation, the data in the warehouse will be as good as the data in the source operational systems (which should utilize RI and check constraints).

ENCOURAGE PARALLELISM

Use partitioned tablespaces and specify 'DEGREE(ANY)' to encourage I/O, CPU, and Sysplex parallelism. Parallelism helps to reduce overall elapsed time when accessing large databases, such as those common in a data warehouse.

Consider partitioning simple and segmented tablespaces to take advantage of DB2's parallelism features. Additionally, consider repartitioning partitioned tablespaces to take full advantage of DB2 parallelism based on the usage patterns of your data warehouse access.

CONSIDER DATA COMPRESSION

As of DB2 Version 3, data compression can be specified directly in a tablespace. Compression is indicated in the DDL by specifying 'COMPRESS YES' for the tablespace. Likewise, it can be turned off in the DDL by specifying 'COMPRESS NO'. When compression is specified, DB2 builds a static dictionary to control compression. It saves from two to 17 dictionary pages in the tablespace. These pages are stored after the header and first space map page.

DB2's hardware-based data compression techniques are optimal for the data warehousing environment. Consider compressing tables that are infrequently accessed to save disk space. Furthermore, consider compressing all tables if possible.

BACK UP THE DATA WAREHOUSE

Putting in place a back-up and recovery plan for data warehouses is imperative. Even though most of the data comes from operational systems originally, you cannot always rebuild data warehouses in the event of a media failure (or a disaster). As operational data ages, it is removed from the operational databases, but it may still exist in the

data warehouse. Furthermore, data warehouses often contain external data that, if lost, may have to be purchased again (creating a financial drain).

FOLLOW 'THE 10 STEPS TO CLEAN DATA'

The following list is a short compendium of the top 10 things you can do to ensure data quality in your data warehouse environment:

1 Foster an understanding of the value of data and information within the organization. In short, treat data as a corporate asset. What does this mean? Consider the other assets of your organization. The capital assets ($) are modelled using a chart of accounts. Human resources (personnel) are modelled using management structures, reporting hierarchies, and personnel files. From building blueprints to item bills of material, every asset that is truly treated as an asset is modelled. If your corporation does not model data, it does not treat data as an asset and is at a disadvantage.

   Acceptance of these ideals can be accomplished through lobbying the users and managers you know, starting an internal newsletter, circulating relevant articles and books throughout your company, and treating data as a corporate asset yourself. A great deal of salesmanship, patience, politics, and good luck will be required, so be prepared.

2 Never cover up data integrity problems. Document them and bring them to the attention of your manager and the users who rely on the data. Usually, the business units using the data are empowered to make changes to it.

3 Do not underestimate the amount of time and effort that will be required to clean up dirty data. Understand the scope of the problem and the process required to rectify it. Take into account the politics of your organization and the automated tools that are available. The more political the battle, the longer the task will take. The fewer tools available, the longer the task will be. Even if you have tools, if no one understands them properly, the situation will probably be worse than having no tools at all, as people struggle to use what they do not understand.

4 Understand what is meant by 'data warehouse' within the context of your projects. What is the scope of the 'warehouse' – enterprise or departmental? What technology is used? If OLAP is a component of the environment, is it ROLAP or MOLAP?

5 Educate those people implementing the data warehouse by sending them on courses and to industry conferences, purchasing books, and encouraging them to read periodicals. A lack of education has killed many potentially rewarding projects.

6 Physically design the data stores for the data warehouse differently from the similar, corresponding production data stores. For example, the file and table structures, indexes, and clustering sequence should be different in the warehouse because the data access requirements are different.

7 You will often hear that denormalization is desirable in the data warehouse environment, but proceed with caution. Because denormalized data is optimized for data access, and the data warehouse is 'read-only', you might think that denormalization is a natural for this environment. However, the data must be populated into the data warehouse at some point. Denormalized data is still difficult to maintain and should be avoided if performance is to be acceptable.

8 Understand the enabling technologies for data warehousing. Replication and propagation are different technologies with different availability and performance effects on both the production (OLTP) and the warehouse (OLAP) systems.

9 Only after you understand the basics should you delve into the more complex aspects of data warehousing such as implementing an ODS, very large databases, or multi-dimensional databases.

10 Reread steps one to nine whenever you think you are overworked, underpaid, or both!

Data in the warehouse is only as good as the sources from which it was gleaned. Failure to clean dirty data can result in the creation of a data outhouse instead of a data warehouse.

USE GOOD DB2 DATABASE DESIGN TECHNIQUES

Use efficient DB2 DDL design techniques such as you would use with any DB2 database design. This includes using the optimal tablespace type (segmented versus partitioned), locking strategy, dataset closing parameter, etc. Good DB2 database design practices must be followed when implementing DB2 data warehouses.

SUMMARY

Data warehouses can provide organizations with a competitive advantage as users begin to analyse data in conjunction with business trends. After a data warehouse is implemented, you cannot turn back because your users will be hooked, your organization will be more profitable, and you'll have the satisfaction of contributing to the success of the business (and, just maybe, a big pay rise).

Craig S Mullins
VP Operations
PLATINUM Technology (USA)

An extent checker

This tool lists the VSAM DB2 LDSs belonging to a special DB2 instance or subsystem along with their extents in sorted form. A threshold is set internally, depending on requirements, for the number of extents a tablespace or indexspace can have. The tool then recalculates the PRIMARY and SECONDARY, based on a pre-set formula.

The tablespaces and the indexspaces that are above the threshold value are then altered and recovered to their new allocations. To restore it to its previous state, a full image copy of the tablespace is taken before the ALTER UTILITY is initiated.

The tool is capable of handling partitioned and non-partitioned tablespaces and the corresponding indexspaces. When using this tool, the outage time for applications because of tablespaces and indexspaces hitting the 119 extents of the DB2 limit for the LDS is minimal.

EXTENT INFO

The EXTENT INFO comprises four components. These are:

- QUESTION
- ALLOC
- LISTCAT
- EXTENT.

The QUESTION member in VCAT.DBAXXX.REXX is executed first and will, in turn, execute the flow shown in Figure 1.

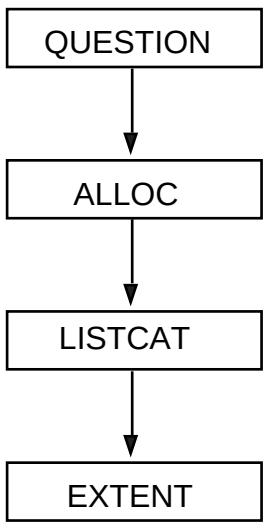

Figure 1: Execution flow

QUESTION

```
/* REXX */
/*****************************************************************/
/* THIS REXX DOES:                                               */
/*                                                               */
/* 1. A QUERY ON SYSSTOGROUP AND GETS VCAT NAME                  */
/* 2. DELETES THE OUTPUT DATASET                                 */
/* 3. EXECUTES THE LISTCAT REXX                                  */
/*                                                               */
```

```
/*****************************************************************/

"ALLOC DS(VCAT.LISTCAT.JCL)  FI(OUT) SHR"
NEWSTACK
QUEUE "//PSYSDBAS  JOB (SYS00000000,B4B),DBAS-DBAXX-SHOP,CLASS=6,    "
QUEUE "//    MSGCLASS=X,NOTIFY=&SYSUID,REGION=2048K,MSGLEVEL=(1,1)  "
QUEUE "//*************************************************************"
QUEUE "//*                                                           "
QUEUE "//* JCL TO DO BACKGROUND DYNAMIC DB2 EXECUTION                "
QUEUE "//*************************************************************"
QUEUE "//STEP010 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)          "
QUEUE "//STEPLIB DD DSN=SYS2.DB2.DBAP.DSNLOAD,DISP=SHR              "
QUEUE "//SYSTSPRT DD SYSOUT=*                                       "
QUEUE "//SYSPRINT DD SYSOUT=*                                       "
QUEUE "//SYSOUT   DD SYSOUT=*                                       "
QUEUE "//SYSREC00 DD DSN=VCAT.DBA.LISTCAT.DATA,DISP=SHR             "
QUEUE "//SYSPUNCH DD DUMMY                                          "
QUEUE "//SYSTSIN  DD *                                              "
QUEUE "    DSN SYSTEM(DBAP)                                         "
QUEUE "   RUN PROGRAM(DSNTIAUL) PARMS('SQL')                        "
QUEUE "   END                                                       "
QUEUE "/*                                                           "
QUEUE "//SYSIN    DD *                                              "
QUEUE "   SET CURRENT DEGREE = 'ANY' ;                              "
QUEUE "   SELECT DISTINCT(VCATNAME) FROM SYSIBM.SYSSTOGROUP;        "
QUEUE "/*                                                           "
QUEUE "//STEP020D EXEC PGM=IEFBR14,COND=(4,LT)                      "
QUEUE "//*                                                          "
QUEUE "//*  - - - - - - - - - - - - - - - - - - - - - - - - - - - - "
QUEUE "//*    SAFETY SCRATCH                                        "
QUEUE "//*  - - - - - - - - - - - - - - - - - - - - - - - - - - - - "
QUEUE "//*                                                          "
QUEUE "//DD01     DD DSN=VCAT.LISTCAT.OUT,                          "
QUEUE "//            DISP=(MOD,DELETE,DELETE),                      "
QUEUE "//            SPACE=(TRK,(0,0),RLSE)                         "
QUEUE "//*                                                          "
QUEUE "//DD02     DD DSN=VCAT.LISTCAT.SORTED,                       "
QUEUE "//            DISP=(MOD,DELETE,DELETE),                      "
QUEUE "//            SPACE=(TRK,(0,0),RLSE)                         "
QUEUE "//*                                                          "
QUEUE "//STEP30   EXEC PGM=IKJEFT01,REGION=4096K                    "
QUEUE "//SYSEXEC  DD DSN=VCAT.DBAXX.REXX,DISP=SHR                   "
QUEUE "//SYSTSPRT DD SYSOUT=*                                       "
QUEUE "//SYSTSIN DD *                                               "
QUEUE "   EXECUTIL SEARCHDD(YES)                                    "
QUEUE "   %ALLOC                                                    "
QUEUE "//*                                                          "
QUEUE "//STEP40   EXEC PGM=IKJEFT01,REGION=4096K                    "
QUEUE "//SYSEXEC  DD DSN=VCAT.DBAXX.REXX,DISP=SHR                   "
QUEUE "//SYSTSPRT DD SYSOUT=*                                       "
```

```
QUEUE "//SYSTSIN DD *                                                  "
QUEUE "   EXECUTIL SEARCHDD(YES)                                       "
QUEUE "   %LISTCAT                                                     "
QUEUE "//*                                                             "
"EXECIO 5Ø DISKW OUT (FINIS"
"SUBMIT 'VCAT.DBA.LISTCAT.JCL'"
"FREE DD(OUT)"
DELSTACK
```

ALLOC

```
/* REXX */
/*********************************************************************/
/* THIS WILL ALLOCATE THE EXTENT OUTPUT DATASET AND THE SORTED OUTP */
/* THE OUTPUT DATASETS ARE ALL PARTITION DATASETS.                   */
/*                                                                   */
/* ASSUMPTION:    THE LIKE DATASET NEEDS TO EXIST BEFORE THIS        */
/*                REXX IS EXECUTED.                                  */
/*********************************************************************/

"ALLOCATE DATASET('VCAT.LISTCAT.OUT') ,
 LIKE('VCAT.LISTCAT.SAMPLE')"
"ALLOCATE DATASET('VCAT.LISTCAT.SORTED') ,
 LIKE('VCAT.LISTCAT.SAMPLE')"
```

LISTCAT

```
/* REXX */
/*********************************************************************/
/*                                                                   */
/* THIS REXX TAKES THE VCATNAME DOWNLOAD AS THE INPUT AND GETS THE    */
/* EXTENT INFORMATION ON ALL THE DSNDBD CLUSTERS WHICH ARE PRESENT IN */
/* THAT CATALOG AND THEN SORTS THEM IN DESCENDING ORDER               */
/*                                                                   */
/*                                                                   */
/* ASSUMPTIONS: 1 THE CATALOG-NAME(VCATS) IS 8 CHARACTERS IN LENGTH   */
/*              2 THE DATABASE NAME IS 8 CHARACTERS IN LENGTH         */
/*              3 THE SPACENAME IS 8 CHARACTERS IN LENGTH             */
/*********************************************************************/

/* REASON FOR USING SORT STEP :                                      */
/*                                                                   */
/* THE ALTER REXX THAT USES THE EXTENT NUMBER IS TAILORED TO DO ALTER */
/* FOR THOSE DATABASE AND SPACENAME WHOSE EXTENT IS > 1ØØ             */
/* (THRESHOLD VALUE)                                                 */
/* HENCE A SORT IS INTRODUCED, WHICH SORTS THE EXTENT NUMBER IN       */
/* ORDER AND PUTS IN A SORTED OUTPUT DATASET                         */
/* THE EXTENT NUMBER POSITION IS TAKEN FROM 45 POSITION BECAUSE OF    */
```

```
/* THE VCATNAME,DATABASE, AND SPACENAM ASSUMPTIONS                    */
/* A SAMPLE OUTPUT IS ATTACHED BELOW                                  */
/* READ THE VCAT DOWNLOAD OF THE PREVIOUS JOB                         */
INDD1="'VCAT.DBA.LISTCAT.DATA'"
"ALLOC DD(IN1) DSN("INDD1") SHR REUSE"
"EXECIO * DISKR IN1  (STEM INPUT. FINIS"
"FREE DD(IN1)"
/* THE LOOP EXECUTES FOR EACH VCAT NAME                               */
DO I=1 TO INPUT.Ø
/* THE SUBSTRING IS DONE TO GET THE EIGHT CHARACTER CATALOG NAME      */
 VCATS = SUBSTR(INPUT.I,1,7)
/* QUEUE THE JCL FOR SUBMISSION                                       */
 "ALLOC DS(DBA.LISTCAT)  FI(OUT) SHR"
 NEWSTACK
 QUEUE "//PSYSDBAS  JOB (SYSØØØØØØØ,B4B),'DBAS-DBAXX-SHOP',CLASS=6,"
 QUEUE "// MSGCLASS=X,REGION=4Ø96K,MSGLEVEL=(1,1),NOTIFY=&SYSID       "
 QUEUE "//JS1Ø     EXEC PGM=IDCAMS                                  "
 QUEUE "//SYSPRINT DD  DSN=VCAT.LISTCAT.DATA,DISP=SHR               "
 QUEUE "//SYSIN    DD  *                                            "
 QUEUE "    LISTCAT -                                               "
 QUEUE "    CATALOG("VCATS") DATA ALL                               "
 QUEUE "/*                                                          "
 QUEUE "//*                                                         "
 QUEUE "//STEPØ2Ø EXEC PGM=IKJEFTØ1,PARM='EXTENT "VCATS"'           "
 QUEUE "//SYSTSPRT DD SYSOUT=*                                      "
 QUEUE "//SYSTSIN  DD DUMMY                                         "
 QUEUE "//SYSEXEC DD DSN=VCAT.DBAXX.REXX,DISP=SHR                   "
 QUEUE "//*                                                         "
 QUEUE "//*****************************************************     "
 QUEUE "//STEPØ3Ø  EXEC PGM=SORT,PARM='SIZE=MAX'                    "
 QUEUE "//*****************************************************     "
 QUEUE "//SYSOUT   DD SYSOUT=*                                      "
 QUEUE "//SORTWKØ1 DD SPACE=(CYL,(5Ø,15)),UNIT=SYSDA                "
 QUEUE "//SORTWKØ2 DD SPACE=(CYL,(5Ø,15)),UNIT=SYSDA                "
 QUEUE "//SORTWKØ3 DD SPACE=(CYL,(5Ø,15)),UNIT=SYSDA                "
 QUEUE "//SORTWKØ4 DD SPACE=(CYL,(5Ø,15)),UNIT=SYSDA                "
 QUEUE "//SORTIN   DD DSN=VCAT.LISTCAT.OUT("VCATS"),DISP=SHR        "
 QUEUE "//SORTOUT  DD DSN=VCAT.LISTCAT.SORTED("VCATS"),DISP=SHR     "
 QUEUE "//SYSIN    DD *                                             "
 QUEUE "     SORT FIELDS=(45,17,CH,D)                               "
 QUEUE "/*                                                          "
 QUEUE "//*                                                         "
 "EXECIO 28 DISKW OUT (FINIS"
 "SUBMIT 'VCAT.DBA.LISTCAT'"
 "FREE DD(OUT)"
 DELSTACK
END
```

LISTCAT SAMPLE OUTPUT

```
/*********************************************************************/
/* SAMPLE OUTPUT                                                     */
/*                                                                   */
/*            CLUSTER NAME                              EXTENT-INFO  */
/* VCATXXXX.DSNDBD.DBXXXXXX.TSXXXXXX.I0001.A001 ---------1           */
/*********************************************************************/
```

EXTENT

```
/*  REXX  */
/*********************************************************************/
/*                                                                   */
/* THIS REXX PICKS UP THE CLUSTER NAME AND THE CORRESPONDING EXTENT*/
/* INFORMATION FROM THE LIST CATALOG OUTPUT AND DUMPS THE CLUSTER  */
/* NAME ALONG WITH THE EXTENT USED IN A OUTPUT FILE                */
/*                                                                   */
/*                                                                   */
/* ASSUMPTIONS : 1. THE VCATNAME IS 8 CHARACTERS IN LENGTH         */
/*               2. THE DATABASE NAME IS 8 CHARACTERS IN LENGTH    */
/*               3. THE SPACENAM IS 8 CHARACTERS IN LENGTH         */
/*********************************************************************/

/* ====> GET THE KEY VALUE PASSED FROM THE PREVIOUS REXX UTIL      */
 ARG KEY
/* ====> ASSIGN THE INPUT LISTCAT INFORMATION FILE                 */
 INDD1="'VCATS.LISTCAT.DATA'"
/* ====> COPY THE INPUT FILE CONTENTS TO STEM VARIABLE IN          */
/*       SHARE REUSE MODE AND FREE THE INPUT FILE                  */
"ALLOC DD(IN1) DSN("INDD1") SHR REUSE"
"EXECIO * DISKR IN1  (STEM INLIST. FINIS"
"FREE DD(IN1)"
/* ====> INTIALIZATION OF THE LOOP INCREMENT                       */
 I=0
/* ====> THE LOOP STARTS READING RECORD BY RECORD FROM THE STEM    */
 DO J=1 TO INLIST.0
/* ====> IF CONDITION TO GET THE CLUSTER PART                      */
/* ===>  REFER TO BELOW OUTPUT                                     */
IF INDEX(INLIST.J,'0DATA ------') ¬=0 THEN DO
/* ====> PARSE THE OTHER VALUES OF THE RECORD AND GET ONLY THE     */
/*       CLUSTER PART INTO REQUAL VARIABLE                         */
      PARSE VALUE INLIST.J WITH DATA '------' REQVAL REST
/* ====> INCREMENT THE COUNTER VARIABLE AND PUT THE REQUAL VALUE   */
/*       INTO THE OUTPUT STEM                                      */
      I=I+1 ;OUTLIST.I= REQVAL
/* ====> THE CONDITION ENDS                                        */
    END
/* ====> IF CONDITION TO GET THE EXTENT INFORMATION FOR THE        */
/*       CLUSTER PART FROM THE PREVIOUS STEPS                      */
```

```
/* ====> REFER TO OUTPUT BELOW
    IF INDEX(INLIST.J,'EXTENTS——') ¬=Ø THEN DO
/* ====> PARSE THE OTHER VALUES OF THE RECORD AND GET ONLY THE     */
/*       EXTENT PART INTO VAL VARIABLE                             */
      PARSE VALUE INLIST.J WITH JUNK1 'EXTENTS' VAL
/* ====> COMBINING THE EXTENT INFORMATION FOR THE CLUSTER GOT      */
      OUTLIST.I= OUTLIST.I  VAL
/* ====> THE CONDITION ENDS                                        */
    END
/* ====> THE DO LOOP ENDS                                         */
 END
/* ====> INITIALIZE VARIABLE FOR NEXT LOOP                        */
 L=Ø
/* ====> FILTERING DSNDB DATA COMPONENT CLUSTERS FROM OTHERS       */
/*                                                                 */
/* ====> ASSIGNING VALUE TO CHECK VARIABLE                        */
 CHKVAR = '.DSNDBD'
/* ====> DO LOOP BEGINS                                           */
DO K=1 TO I
/* ====> CHECK FOR THE DATA COMPONENT                             */
/* THE SUBSTRING IS DONE TO CHECK WHETHER THE ".DSNDBD" PART       */
/* FALLS IN THE EIGHTH POSTION OF THE CLUSTER NAME ACCORDING       */
/* TO THE ASSUMPTIONS.                                            */
    IF SUBSTR(OUTLIST.K,8,7) == CHKVAR THEN  DO
/* ====> WRITING TO THE OUTPUT STEM IF CONDITION SATISFIES         */
        L=L+1 ;OUT.L=OUTLIST.K
/* ====> THE IF CONDITION ENDS                                    */
    END
/* ====> THIS DISPLAYS THE OTHER CLUSTERS AND THEIR EXTENTS        */
    ELSE DO
       SAY  OUTLIST.K
    END
/* THE DO LOOP ENDS                                               */
 END
/* ====> ASSIGN THE OUTPUT FILE                                   */
OUTDS="'VCAT.LISTCAT.OUT("KEY")'"
/* ====> WRITE THE CONTENTS OF THE OUTPUT STEM INTO THE FILE      */
"ALLOC DD(ODD) DSN("OUTDS") SHR "
"EXECIO * DISKW ODD  (STEM  OUT. FINIS"
/* ====> FREE OUTPUT FILE                                         */
"FREE DD(ODD)"
/* ====> THE REXX ENDS HERE                                       */
EXIT
```

EXTENT SAMPLE OUTPUT

```
/*****************************************************************/
/* SAMPLE OUTPUT                                                 */
/*                                                               */
```

```
/* LISTCAT OUTPUT - :
1IDCAMS  SYSTEM SERVICES                                        TIME:
Ø
     LISTCAT -
     CATALOG(SYSDBQP) DATA ALL
1IDCAMS  SYSTEM SERVICES                                        TIME:
-                              LISTING FROM CATALOG — SYSDBQP
ØDATA ————— VCATNAME.DBXXXXXX.TSXXXXXX.I0000.A001
   HISTORY
    DATASET-OWNER——(NULL)     CREATION———1994.177
    RELEASE————————2     EXPIRATION——0000.000
    ACCOUNT-INFO————————————————(NULL)
    PROTECTION-PSWD——(NULL)     RACF————————(NO)
    ASSOCIATIONS
     CLUSTER—VCATNAME.DBXXXXXX.TSXXXXXX.I0000.A001
    ATTRIBUTES
     KEYLEN————————4  AVGLRECL———————4089 BUFSPA
     RKP————————————Ø  MAXLRECL———————4089 EXCPEX
     SHROPTNS(2,3)   RECOVERY  UNIQUE           NOERASE INDEXE
     UNORDERED    TEMP-EXP  REUSE   NONSPANNED EXTENTS————————132
*/
```

RUNNING THE EXEC

The following are the steps to run the EXEC:

- Allocate a PDS dataset ‘xxxTEST.USERID.REXXLIB’ of LRECL 80 and copy the three EXECS EXTENTS, ALTERIX, and ALTERTS.
- Allocate SEQ datasets:
  - ‘xxxTEST.USERID.INPUT’ of LRECL 80 which will have the input LDSs and EXTENTS, sorted on the extents.
  - ‘xxxTEST.USERID.JCLOUT’ of LRECL 80 where the JCL will be QUEUED.
  - ‘xxxTEST.USERID.JCLOUT.DATA’ in which the records are unloaded. The LRECL is set depending on the length of the record unloaded.
  - ‘xxxTEST.USERID.ALTER.JCL’ of LRECL 80 where the JCL to ALTER is QUEUED.
- Executing the REXX in ‘xxxTEST.USERID. REXXLIB(EXTENTS)’ will start the utility.

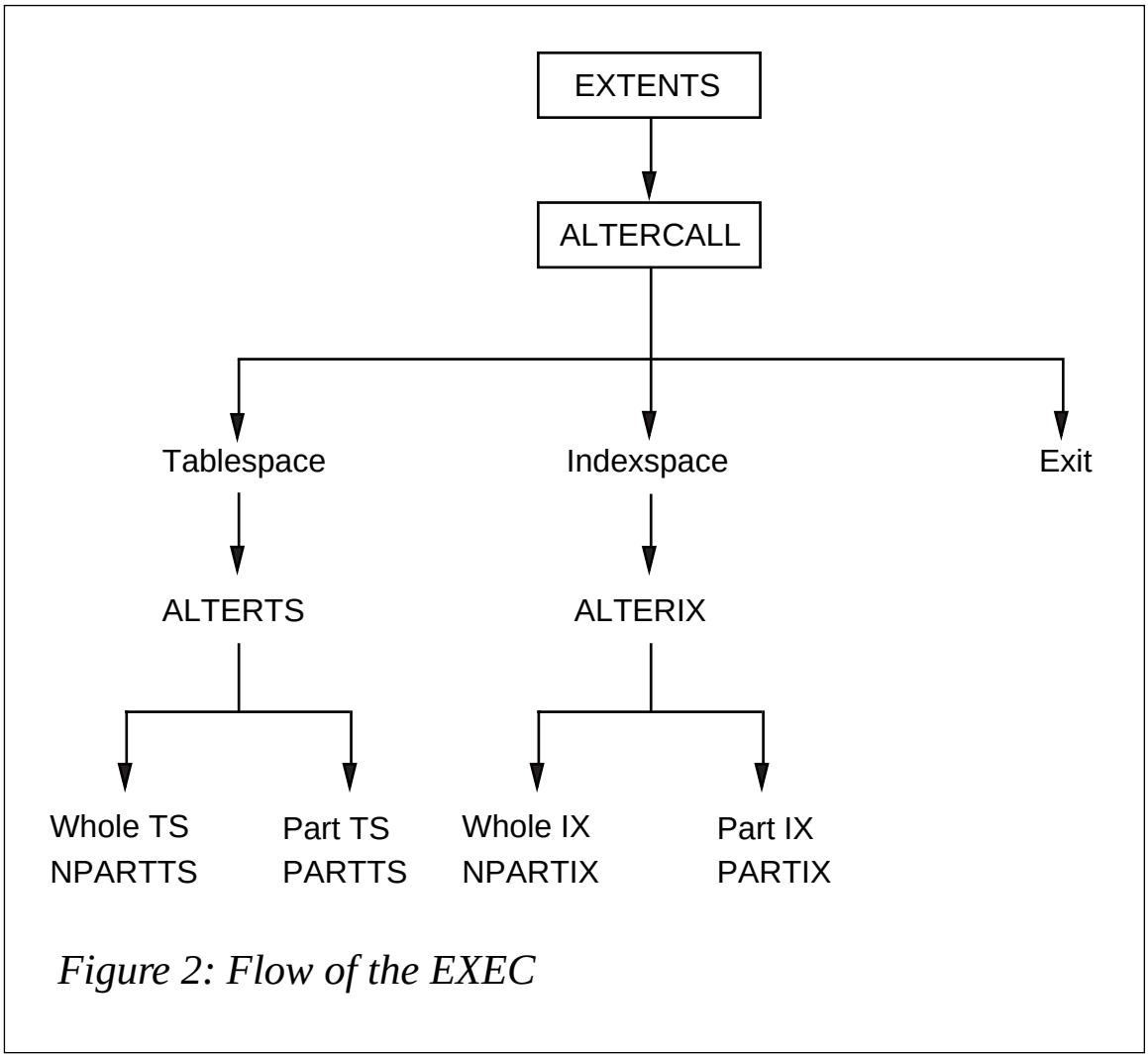

Figure 2: Flow of the EXEC

The flow of the EXEC is shown in Figure 2.

EXTENTS

```
/* REXX TO GET THE INCREASED PQTY AND SECQTY        */
/* ASSUMES THE FILE XXXTEST.USERID.INPUT IS         */
/*  ALLOCATED                                       */
   "ALLOC DS(USERID.INPUT)  FI(INP) SHR REU"
/* INITIALIZE THE VARIABLES                         */
   DSNAME. = ''
   I = Ø
/* READ THE FILE INTO AN ARRAY                      */
   "EXECIO * DISKR INP (STEM DSNAME."
   ENDFILE = DSNAME.Ø

/* THE DATA IS ASSUMED TO BE IN THE FORMAT AS GIVEN */
```

```
/*  BELOW AND SORTED ON THE EXTENTS :               */
/* XXXTEST.DSNDBD.DATABASE.SPACENAM.I0001.AZZZ NUM  */
/*  WHERE XXXTEST  - VCATNAME OF THE STGROUP (VCAT) */
/*        DATABASE - DATABASE NAME OF 8 CHARS (DB)  */
/*        SPACENAM - SPACENAME OF 8 CHARS (SP)      */
/*        NUM      - NUMBER OF EXTENTS (EXT)        */
/*        ZZZ      - USED FOR THE PARTITION (PART)  */
  DO I = I + 1 UNTIL I >= ENDFILE
   PARSE VALUE DSNAME.I WITH VT '.' X '.' DB '.' SP '.' Y ' ' EXT ' ' Z
   PART = SUBSTR(Y,8,3)

/* DISPLAY THE PARAMETERS                           */
   SAY 'DB NAME : ' DB    'SPACENM : ' SP
   SAY 'PART    : ' PART  'EXTENTS : ' EXT

/* CHECK IF THE EXTENTS HAVE EXCEEDED THE LIMIT     */
/*     AFTER WHICH YOU WANT TO ALTER THE SPACE      */
     IF (EXT > LIMIT)
        THEN
           CALL ALTERCALL
     ELSE
        DO
           SAY 'END OF THE ALTER FOR THIS VCAT' VT
           EXIT
        END
  END
EXIT

/* END OF THE REXX                                  */
/* ROUTINE TO CHECK WHETHER THE SPACENAME IS        */
/*  A TABLESPACE OR AN INDEXSPACE                   */
/* THE 4TH CHAR OF THE SPACENAME IS THE ONE WHICH   */
/*  SEPARATES THE TABLESPACES FROM THE INDEXSPACES  */

ALTERCALL:
 SPACE_NM = SUBSTR(SP,4,1)
   IF SPACE_NM = 'I'
      THEN
        DO
         SAY 'INDEXSPACE IS TO BE ALTERED'
         CALL ALTERIX
        END
   ELSE
   IF SPACE_NM = 'S'
      THEN
        DO
         SAY 'TABLESPACE IS TO BE ALTERED'
         CALL ALTERTS
        END
   ELSE
```

```
         SAY 'INVALID SPACE NAME IN THE INPUT' SP
RETURN

/* ROUTINE TO GET THE INCREASED PQTY AND SECQTY FOR */
/* THE PARTICULAR INDEXSPACE OR A PART OF THE       */
/* INDEXSPACE.                                      */
/*   THE PQTY AND THE SECQTY IS PUMPED BY 30% AND   */
/*     20% RESPECTIVELY                             */

ALTERIX:
/* QUEUE THE JCL AND SUBMIT IT                      */
"ALLOC DS(USERID.JCLOUT)  FI(OUT) SHR"
NEWSTACK
QUEUE "//XXXXXJOB  JOB (ACCT PARAMETER),'COMMENTS',CLASS=A,         "
QUEUE "//  MSGCLASS=X,REGION=4096K,MSGLEVEL=(1,1),NOTIFY=&SYSUID    "
QUEUE "//*************************************************************"
QUEUE "//* TO GET THE PQTY AND SQTY OF THE INDEXSPACE               "
QUEUE "//*************************************************************"
QUEUE "//STEP010 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)          "
QUEUE "//STEPLIB DD DSN=SYS2.DB2.XXXX.DSNLOAD,DISP=SHR              "
QUEUE "//SYSTSPRT DD SYSOUT=*                                       "
QUEUE "//SYSPRINT DD SYSOUT=*                                       "
QUEUE "//SYSOUT   DD SYSOUT=*                                       "
QUEUE "//SYSREC00 DD DSN=XXXTEST.USERID.JCLOUT.DATA,DISP=SHR        "
QUEUE "//SYSPUNCH DD DUMMY                                          "
QUEUE "//SYSTSIN  DD *                                              "
QUEUE "    DSN SYSTEM(XXXX)                                         "
QUEUE "   RUN PROGRAM(DSNTIAUL) PARMS('SQL')                        "
QUEUE "   END                                                       "
QUEUE "/*                                                           "
IF PART = 1
 THEN
  DO
    QUEUE "//SYSIN    DD *                                          "
    QUEUE "   SET CURRENT DEGREE = 'ANY' ;                          "
    QUEUE "   SELECT IXCREATOR, '"DB"', ' ', '"SP"', ' ',           "
    QUEUE "          DIGITS(PARTITION), ' ',                        "
    QUEUE "          DIGITS(INTEGER(PQTY * 1.2) + (PQTY * 4)), ' ', "
    QUEUE "          DIGITS(INTEGER(SQTY * .8) + (SQTY * 4))        "
    QUEUE "   FROM SYSIBM.SYSINDEXPART                              "
    QUEUE "   WHERE IXNAME = '"SP"' AND PARTITION IN(0,1);          "
    QUEUE "/*                                                       "
  END
ELSE
  DO
    QUEUE "//SYSIN    DD *                                          "
    QUEUE "   SET CURRENT DEGREE = 'ANY' ;                          "
    QUEUE "   SELECT IXCREATOR, '"DB"', ' ', '"SP"', ' ',           "
    QUEUE "          DIGITS(PARTITION), ' ',                        "
    QUEUE "          DIGITS(INTEGER(PQTY * 1.2) + (PQTY * 4)), ' ', "
```

```
    QUEUE "          DIGITS(INTEGER(SQTY * .8) + (SQTY * 4))        "
    QUEUE "   FROM SYSIBM.SYSINDEXPART                              "
    QUEUE "   WHERE IXNAME = '"SP"' AND PARTITION = "PART";         "
    QUEUE "/*                                                       "
  END
QUEUE "//*************************************************************"
QUEUE "//* TO EXEC THE ALTER                                        "
QUEUE "//*************************************************************"
QUEUE "//STEP2Ø   EXEC PGM=IKJEFTØ1,REGION=4Ø96K                    "
QUEUE "//SYSEXEC  DD DSN=XXXTEST.USERID.REXXLIB,DISP=SHR            "
QUEUE "//SYSTSPRT DD SYSOUT=*                                       "
QUEUE "//SYSTSIN DD *                                               "
QUEUE "   EXECUTIL SEARCHDD(YES)                                    "
QUEUE "   %ALTERIX                                                  "
 QUEUE "/*                                                          "
"EXECIO 36 DISKW OUT (FINIS"
"SUBMIT 'XXXTEST.USERID.JCLOUT'"
"FREE DD(OUT)"
DELSTACK
RETURN

/* ROUTINE TO GET THE INCREASED PQTY AND SECQTY FOR */
/* THE PARTICULAR TABLESPACE OR A PART OF THE       */
/* TABLESPACE                                       */
/*   THE PQTY AND THE SECQTY IS PUMPED BY 3Ø% AND   */
/*     2Ø% RESPECTIVELY                             */

ALTERTS:
/* QUEUE THE JCL AND SUBMIT IT                      */
"ALLOC DS(USERID.JCLOUT)  FI(OUT) SHR"
NEWSTACK
QUEUE "//XXXXXJOB  JOB (ACCT PARAMETER),'COMMENTS',CLASS=A,         "
QUEUE "//  MSGCLASS=X,REGION=4Ø96K,MSGLEVEL=(1,1),NOTIFY=&SYSUID    "
QUEUE "//*************************************************************"
QUEUE "//* TO GET THE PQTY AND SQTY OF THE TABLESPACE               "
QUEUE "//*************************************************************"
QUEUE "//STEPØ1Ø EXEC PGM=IKJEFTØ1,DYNAMNBR=2Ø,COND=(4,LT)          "
QUEUE "//STEPLIB DD DSN=SYS2.DB2.XXXX.DSNLOAD,DISP=SHR              "
QUEUE "//SYSTSPRT DD SYSOUT=*                                       "
QUEUE "//SYSPRINT DD SYSOUT=*                                       "
QUEUE "//SYSOUT   DD SYSOUT=*                                       "
QUEUE "//SYSREC ØØ DD DSN=XXXTEST.USERID.JCLOUT.DATA,DISP=SHR       "
QUEUE "//SYSPUNCH DD DUMMY                                          "
QUEUE "//SYSTSIN  DD *                                              "
QUEUE "    DSN SYSTEM(XXXX)                                         "
QUEUE "   RUN PROGRAM(DSNTIAUL) PARMS('SQL')                        "
QUEUE "   END                                                       "
QUEUE "/*                                                           "

IF PART = 1
```

```
 THEN
  DO
    QUEUE "//SYSIN    DD *                                                "
    QUEUE "   SET CURRENT DEGREE = 'ANY' ;                                "
    QUEUE "   SELECT '"VT"', ' ', '"DB"', ' ', '"SP"', ' ',               "
    QUEUE "          DIGITS(PARTITION), ' ',                              "
    QUEUE "          DIGITS(INTEGER(PQTY * 1.2) + (PQTY * 4)), ' ',       "
    QUEUE "          DIGITS(INTEGER(SQTY * .8) + (SQTY * 4))              "
    QUEUE "   FROM SYSIBM.SYSTABLEPART                                    "
    QUEUE "   WHERE TSNAME = '"SP"' AND PARTITION IN (Ø,1);               "
    QUEUE "/*                                                             "
  END
 ELSE
  DO
    QUEUE "//SYSIN    DD *                                                "
    QUEUE "   SET CURRENT DEGREE = 'ANY' ;                                "
    QUEUE "   SELECT '"VT"', ' ', '"DB"', ' ', '"SP"', ' ',               "
    QUEUE "          DIGITS(PARTITION), ' ',                              "
    QUEUE "          DIGITS(INTEGER(PQTY * 1.2) + (PQTY * 4)), ' ',       "
    QUEUE "          DIGITS(INTEGER(SQTY * .8) + (SQTY * 4))              "
    QUEUE "   FROM SYSIBM.SYSTABLEPART                                    "
    QUEUE "   WHERE TSNAME = '"SP"' AND PARTITION = "PART";               "
    QUEUE "/*                                                             "
  END
QUEUE "//************************************************************"
QUEUE "//* TO EXEC THE ALTER                                          "
QUEUE "//************************************************************"
QUEUE "//STEP2Ø   EXEC PGM=IKJEFTØ1,REGION=4Ø96K                       "
QUEUE "//SYSEXEC  DD DSN=XXXTEST.USERID.REXXLIB,DISP=SHR               "
QUEUE "//SYSTSPRT DD SYSOUT=*                                          "
QUEUE "//SYSTSIN DD *                                                  "
QUEUE "   EXECUTIL SEARCHDD(YES)                                       "
QUEUE "   %ALTERTS                                                     "
 QUEUE "//*                                                            "
"EXECIO 36 DISKW OUT (FINIS"
"SUBMIT 'XXXTEST.USERID.JCLOUT'"
"FREE DD(OUT)"
DELSTACK
RETURN
```

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

Kiran Haryadi and K R Swaminaathan
DBA
Wipro Infotech (India)

Analysing the DSNZPARM load module – part 2

This month we continue the program that analyses the DSNZPARM load module and creates the originating assembly macro input.

```
*> SEQPRES  - SEQU. UTILITY DATA IN 399Ø  CACHE
         MVC   ZPRMCL16(12),=C'SEQPRES=YES '
         TM    SPRMMIS2,B'Ø1ØØØØØØ'       BIT 1
         BO    *+1Ø
         MVC   ZPRMCL16+Ø8(Ø3),=C'NO '
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN              DO PRINT LINE
         AIF   (NOT D'SPRMPAC).CACHDYN             IF V5 THEN   V5
*> CACHEDYN - CACHE DYNAMIC SQL IN EDM POOL                     V5
         MVC   ZPRMCL16(12),=C'CACHEDYN=YES'                   V5
         TM    SPRMMIS2,B'ØØØ1ØØØØ'       BIT 3               V5
         BO    *+1Ø                                          V5
         MVC   ZPRMCL16(12),=C'CACHEDYN=NO '                   V5
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK     V5
         MVI   Ø(1),C','                  PLUG COMMA HERE      V5
         BAS   R14,ZWRTRTN              DO PRINT LINE          V5
*> CACHEPAC - CACHE FOR PACKAGE AUTHORIZATION                   V5
         MVC   ZPRMCL16(Ø9),=C'CACHEPAC=' FIELD LITERAL        V5
         ICM   R9,15,SPRMPAC              GET ZPARM VALUE      V5
         CVD   R9,D                       CONVERT DECIMAL      V5
         UNPK  ZPRMCL16+Ø9(15),D          PACK TO ZONE         V5
         OI    ZPRMCL16+23,X'FØ'          FIX LAST DIGIT       V5
         MVC   ZEROHOLD,ZPRMCL16+Ø9       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS   V5
         MVC   ZPRMCL16+Ø9(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK     V5
         MVI   Ø(1),C','                  PLUG COMMA HERE      V5
         BAS   R14,ZWRTRTN              DO PRINT LINE          V5
*> MAXKEEPD - DYNAMIC SQL KEPT AFTER COMMIT                     V5
         MVC   ZPRMCL16(Ø9),=C'MAXKEEPD=' FIELD LITERAL        V5
         ICM   R9,15,SPRMMXKD             GET ZPARM VALUE      V5
         CVD   R9,D                       CONVERT DECIMAL      V5
         UNPK  ZPRMCL16+Ø9(15),D          PACK TO ZONE         V5
         OI    ZPRMCL16+23,X'FØ'          FIX LAST DIGIT       V5
         MVC   ZEROHOLD,ZPRMCL16+Ø9       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS   V5
         MVC   ZPRMCL16+Ø9(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK     V5
         MVI   Ø(1),C','                  PLUG COMMA HERE      V5
         BAS   R14,ZWRTRTN              DO PRINT LINE          V5
*> RELCURHL - RELEASE CURSOR WITH HOLD AT COMMIT               V5
```

```
         MVC   ZPRMCL16(12),=C'RELCURHL=NO '                    V5
         TM    SPRMMIS2,B'00001000'       BIT 4                 V5
         BZ    *+10                                             V5
         MVC   ZPRMCL16(12),=C'RELCURHL=YES'                    V5
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK      V5
         MVI   0(1),C','                  PLUG COMMA HERE       V5
         BAS   R14,ZWRTRTN             DO PRINT LINE            V5
.CACHDYN ANOP                                        V5 ELSE   V5
*> SITETYP  - SITE TYPE
         MVC   ZPRMCL16(08),=C'SITETYP='
         TM    SPRMTYP,B'10000000'        BIT 1
         BNO   NOTYP1
         MVC   ZPRMCL16+08(09),=C'LOCALSITE'
         B     YESTYP1
NOTYP1   MVC   ZPRMCL16+08(03),=C'NO '
YESTYP1  TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   0(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> SRTPOOL  - SORT POOL
         MVC   ZPRMCL16(08),=C'SRTPOOL=' LITERAL
         SR    R9,R9                      ZERO REGISTER
         L     R8,SPRMSORP
         SRDA  R8,32(0)                   SHIFT RIGHT 32 BITS
         D     R8,=F'4096'                DIVIDE BY 4096
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+08(15),D
         OI    ZPRMCL16+22,X'F0'
         MVC   ZEROHOLD,ZPRMCL16+08       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+08(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   0(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> SYSADM  - SYSTEM ADMINISTRATOR 1
         MVC   ZPRMCL16(07),=C'SYSADM='
         MVC   ZPRMCL16+07(08),SPRMSADM
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   0(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> SYSADM2 - SYSTEM ADMINISTRATOR 2
         MVC   ZPRMCL16(08),=C'SYSADM2='
         MVC   ZPRMCL16+08(08),SPRMADM2
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   0(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> SYSOPR1 - SYSTEM OPERATOR 1
         MVC   ZPRMCL16(08),=C'SYSOPR1='
         MVC   ZPRMCL16+08(08),SPRMOPR1
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   0(1),C','                  PLUG COMMA HERE
```

```
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> SYSOPR2 - SYSTEM OPERATOR 2
         MVC   ZPRMCL16(Ø8),=C'SYSOPR2='
         MVC   ZPRMCL16+Ø8(Ø8),SPRMOPR2
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> UTIMOUT - UTILITY TIMEOUT FACTOR
         MVC   ZPRMCL16(Ø8),=C'UTIMOUT='  FIELD LITERAL
         SR    R9,R9                      ZERO REGISTER
         LH    R9,SPRMUTO                 GET ZPARM VALUE
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø8(7),D           PACK TO ZONE NUMERIC
         OI    ZPRMCL16+14,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø8       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø8(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         MVI   ZPRMCL72,C' '              PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
         TITLE 'LOEBEN - ZPARM DB2 V4 CREATE - 27.1Ø.98                *
               DSN6ARVP                    '
*> FORMAT DSN6ARVP ************************************************
         USING DSN6ARVP,R7
         L     R7,ZPARMPTR
         LA    RØ,4
         LA    R1,255(,R7)
         CLC   =CL8'DSN6ARVP',4(R7)
         BE    *+12
         BXLE  R7,RØ,*-1Ø
         B     ABEND192
         L     R7,Ø(,R7)
         L     R2,=A(DSN6ARVP)            SECTION TO BE ANALYSED
         CLC   ARVPID,ARVPID-DSN6ARVP(R2)
         BNE   ABEND1Ø3
         CLC   ARVPEID,ARVPEID-DSN6ARVP(R2)
         BNE   ABEND1Ø3                   SECTION DSN6ARVP NOT FOUND
         MVC   ZPRMCLØ5(Ø8),=C'DSN6ARVP'
*> ALCUNIT - ARCHIVE ALLOCATION UNIT
         MVC   ZPRMCL16(11),=C'ALCUNIT=CYL'
         MVC   WRKPFLG1,ARVPFLG1          SAVE
         TM    ARVPFLG1,B'Ø1ØØØØØØ'       CYL ?
         BO    FLG145                     Y
         MVC   ZPRMCL16+Ø8(Ø3),=C'TRK'
         TM    ARVPFLG1,B'ØØ1ØØØØØ'       TRK ?
         BO    FLG145                     Y
         MVC   ZPRMCL16+Ø8(Ø3),=C'BLK'    DEFAULT IS BLK
FLG145   TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> ARCWRTC - ARCHIVE WRITE ROUTE CODE
```

```
         MVC   ZPRMCL16(Ø8),=C'ARCWRTC='  FIELD LITERAL
         MVC   WORKB16,ARVPWTØ1+X'88'
         BAS   R14,BIT16RTN
         CLI   WORKCHR1,C')'              IF ) MEANS ALL BITS ARE Ø
         BNE   *+1Ø                       N. GO ON
         MVC   WORKCHAR(Ø2),=C'NO'        Y. SAY NO HERE
         MVC   ZPRMCL16+Ø8(48),WORKCHAR
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> ARCWTOR - ARCHIVE WRITE TO OPERATOR REPLY
         MVC   ZPRMCL16(11),=C'ARCWTOR=YES'
         TM    ARVPFLG1,B'ØØØØ1ØØØ'       BIT 5 ON
         BO    *+1Ø                       Y.
         MVC   ZPRMCL16+Ø8(Ø3),=C'NO '
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> ARCPFX1 - ARCHIVE PREFIX NAME 1
         MVC   ZPRMCL16(Ø8),=C'ARCPFX1='  FIELD LITERAL
         MVC   ZPRMCL16+Ø8(35),ARVPRE1N   GET ZPARM VALUE
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> ARCPFX2 - ARCHIVE PREFIX NAME 2
         MVC   ZPRMCL16(Ø8),=C'ARCPFX2='  FIELD LITERAL
         MVC   ZPRMCL16+Ø8(35),ARVPRE2N   GET ZPARM VALUE
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> ARCRETN - ARCHIVE RETENTION PERIOD
         MVC   ZPRMCL16(Ø8),=C'ARCRETN='
         SR    R9,R9                      ZERO REGISTER
         LH    R9,ARVPRETN                GET ZPARM VALUE
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø8(7),D           PACK TO ZONE NUMERIC
         OI    ZPRMCL16+14,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø8       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø8(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> BLKSIZE - ARCHIVE BLOCKSIZE
         MVC   ZPRMCL16(Ø8),=C'BLKSIZE='
         SR    R9,R9                      ZERO REGISTER
         L     R9,ARVPBKSZ                GET ZPARM VALUE
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø8(15),D          PACK TO ZONE NUMERIC
         OI    ZPRMCL16+22,X'FØ'          FIX LAST DIGIT
```

```
         MVC   ZEROHOLD,ZPRMCL16+Ø8       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø8(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> CATALOG - CATALOG ARCHIVE DATASET NAME
         MVC   ZPRMCL16(Ø8),=C'CATALOG='
         TM    ARVPFLG1,B'1ØØØØØØØ'       BIT 1 ON
         BNO   NOFLG11                    Y.
         MVC   ZPRMCL16+Ø8(Ø3),=C'YES'    N.
         B     YESFLG11
NOFLG11  MVC   ZPRMCL16+Ø8(Ø3),=C'NO '
YESFLG11 TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> COMPACT - COMPACT ENABLED/DISABLED
         MVC   ZPRMCL16(Ø8),=C'COMPACT='
         TM    ARVPFLG1,B'ØØØØØ1ØØ'       BIT 6 ON
         BNO   NOFLG16                    Y.
         MVC   ZPRMCL16+Ø8(Ø3),=C'YES'    N.
         B     YESFLG16
NOFLG16  MVC   ZPRMCL16+Ø8(Ø3),=C'NO '
YESFLG16 TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> MSVGP   - NAME OF A GROUP OF MSS VOLUMES FOR ARC LOG DS 1
         MVC   ZPRMCL16(Ø6),=C'MSVGP='    FIELD LITERAL
         MVC   ZPRMCL16+Ø6(Ø8),ARVPMSV1   GET ZPARM VALUE
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> MSVGP2  - NAME OF A GROUP OF MSS VOLUMES FOR ARC LOG DS
         MVC   ZPRMCL16(Ø7),=C'MSVGP2='   FIELD LITERAL
         MVC   ZPRMCL16+Ø7(Ø8),ARVPMSV2   GET ZPARM VALUE
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> PRIQTY  - PRIMARY SPACE ALLOCATION
         MVC   ZPRMCL16(Ø7),=C'PRIQTY='
         SR    R9,R9                      ZERO REGISTER
         L     R9,ARVPRISP                GET ZPARM VALUE
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø7(15),D          PACK TO ZONE NUMERIC
         OI    ZPRMCL16+21,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø7       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø7(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
```

```
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> PROTECT - RACF PROTECTION OF ARCHIVE LOG DATA SET
         MVC   ZPRMCL16(Ø8),=C'PROTECT='
         TM    ARVPFLG1,B'ØØØ1ØØØØ'       BIT 4 ON
         BNO   NOFLG14                    Y.
         MVC   ZPRMCL16+Ø8(Ø3),=C'YES'    N.
         B     YESFLG14
NOFLG14  MVC   ZPRMCL16+Ø8(Ø3),=C'NO '
YESFLG14 TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> QUIESCE - MAX QUIESCE PERIOD
         MVC   ZPRMCL16(Ø8),=C'QUIESCE='
         SR    R9,R9                      ZERO REGISTER
         LH    R9,ARVPMQP                 GET ZPARM VALUE
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø8(7),D           PACK TO ZONE NUMERIC
         OI    ZPRMCL16+14,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø8       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø8(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> SECQTY  - SECONDARY SPACE ALLOCATION
         MVC   ZPRMCL16(Ø7),=C'SECQTY='
         SR    R9,R9                      ZERO REGISTER
         L     R9,ARVPSECS                GET ZPARM VALUE
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø7(15),D          PACK TO ZONE NUMERIC
         OI    ZPRMCL16+21,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø7       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø7(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> TSTAMP  - TIME STAMP IN ARCHIVE LOG DATA SET
         MVC   ZPRMCL16(Ø7),=C'TSTAMP='
         TM    ARVPFLG1,B'ØØØØØØ1Ø'       BIT 7 ON
         BNO   NOFLG17                    Y.
         MVC   ZPRMCL16+Ø7(Ø3),=C'YES'    N.
         B     YESFLG17
NOFLG17  MVC   ZPRMCL16+Ø7(Ø3),=C'NO '
YESFLG17 TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> UNIT    - TAPE DEVICE TYPE
         MVC   ZPRMCL16(Ø5),=C'UNIT='     FIELD LITERAL
         MVC   ZPRMCL16+Ø5(Ø8),ARVPUNT1   GET ZPARM VALUE
```

```
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
         MVC   ZPRMCL16(Ø6),=C'UNIT2='    FIELD LITERAL
         MVC   ZPRMCL16+Ø6(Ø8),ARVPUNT2   GET ZPARM VALUE
         MVI   ZPRMCL72,C' '              PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
         TITLE 'LOEBEN - ZPARM DB2 V4 CREATE - 27.1Ø.98               *
               DSN6LOGP                '
*> FORMAT DSN6LOGP    ***********************************************
         USING DSN6LOGP,R7
         L     R7,ZPARMPTR
         LA    RØ,4
         LA    R1,255(,R7)
         CLC   =CL8'DSN6LOGP',4(R7)
         BE    *+12
         BXLE  R7,RØ,*-1Ø
         B     ABEND194
         L     R7,Ø(,R7)
         L     R2,=A(DSN6LOGP)         SECTION TO BE ANALYSED
         CLC   LOGPID,LOGPID-DSN6LOGP(R2)
         BNE   ABEND1Ø4
         CLC   LOGPEID,LOGPEID-DSN6LOGP(R2)
         BNE   ABEND1Ø4                SECTION DSN6LOGP NOT FOUND
         MVC   ZPRMCLØ5(Ø8),=C'DSN6LOGP'
*> DEALLCT  - DEALLOCATION TIME IN MINUTES
         MVC   ZPRMCL16(12),=C'DEALLCT=(Ø) '
         LH    R9,LOGPDMIN                GET ZPARM VALUE
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø9(Ø7),D          PACK TO ZONE NUMERIC
         OI    ZPRMCL16+15,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø9       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø9(Ø7),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVC   Ø(2,R1),=C'),'             PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> INBUFF   - INPUT BUFFER POOL SIZE
         MVC   ZPRMCL16(Ø7),=C'INBUFF='
         SR    R9,R9                      ZERO REGISTER
         L     R9,LOGPIBPS                GET ZPARM VALUE
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø7(15),D          PACK TO ZONE NUMERIC
         OI    ZPRMCL16+21,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø7       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø7(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
```

```
*> MAXARCH  - MAX ARCHIVE ENTRIES IS BSDS
         MVC   ZPRMCL16(Ø8),=C'MAXARCH='
         SR    R9,R9                      ZERO REGISTER
         L     R9,LOGPARCL                GET ZPARM VALUE
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø8(7),D           PACK TO ZONE NUMERIC
         OI    ZPRMCL16+14,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø8       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø8(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> MAXRTU   - MAXIMUM ARCHIVE READ TAPE UNITS
         MVC   ZPRMCL16(Ø7),=C'MAXRTU='
         SR    R9,R9                      ZERO REGISTER
         LH    R9,LOGPMRTU                GET ZPARM VALUE
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø7(7),D           PACK TO ZONE NUMERIC
         OI    ZPRMCL16+13,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø7       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø7(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> OUTBUFF  - OUTPUT BUFFER POOL SIZE
         MVC   ZPRMCL16(Ø8),=C'OUTBUFF='
         SR    R9,R9                      ZERO REGISTER
         L     R9,LOGPOBPS                GET ZPARM VALUE
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø8(15),D          PACK TO ZONE NUMERIC
         OI    ZPRMCL16+22,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø8       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø8(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> TWOACTV  -
         MVC   ZPRMCL16(12),=C'TWOACTV=NO  '
         TM    LOGOPT1,128
         BZ    *+1Ø
         MVC   ZPRMCL16+8(3),=C'YES'
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> TWOARCH  -
         MVC   ZPRMCL16(12),=C'TWOARCH=NO  '
         TM    LOGOPT2,128
```

```
         BZ    *+1Ø
         MVC   ZPRMCL16+8(3),=C'YES'
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> TWOBSDS  -
         MVC   ZPRMCL16(12),=C'TWOBSDS=NO  '
         TM    LOGOPT1,32
         BZ    *+1Ø
         MVC   ZPRMCL16+8(3),=C'YES'
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> WRTHRSH  -
         MVC   ZPRMCL16(Ø8),=C'WRTHRSH='
         SR    R9,R9                      ZERO REGISTER
         LH    R9,LOGPWRTH                GET ZPARM VALUE
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø8(7),D           PACK TO ZONE NUMERIC
         OI    ZPRMCL16+14,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø8       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø8(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         MVI   ZPRMCL72,C' '              PLUG SPACE IN COL 72
         BAS   R14,ZWRTRTN             DO PRINT LINE
         TITLE 'LOEBEN - ZPARM DB2 V4 CREATE - 27.1Ø.98               *
               DSN6SYSP                  '
*> FORMAT DSN6SYSP    ***********************************************
         L     R7,ZPARMPTR
         LA    RØ,4
         LA    R1,255(,R7)
         CLC   =CL8'DSN6SYSP',4(R7)
         BE    *+12
         BXLE  R7,RØ,*-1Ø
         B     ABEND195
         L     R7,Ø(,R7)
         USING DSN6SYSP,R7
         LR    R15,R7
         LA    RØ,1
         LA    R1,DSN6SYSP+L'SYSPLVLC-1
         DROP  R7
         USING DSN6SYSP,R15
         CLI   SYSPLVLC,C' '
         BL    ABEND1Ø5
         BXLE  R15,RØ,*-8
         DROP  R15
         USING DSN6SYSP,R7
*> FORMAT DSN6SYSP    ***********************************************
         MVC   ZPRMCLØ5(Ø8),=C'DSN6SYSP'
*> AUDITST  - AUDIT TRACE START
```

```
         MVC   ZPRMCL16(Ø8),=C'AUDITST='
         MVC   WORKB32,SYSPAUDT           GET 32 BITS
         BAS   R14,BIT16RTN               CONVERT FIRST 16 BITS TO NUM
         BAS   R14,BIT32RTN               CONVERT NEXT 16 BITS TO NUM
         CLI   WORKCHR1,C')'              IF ) MEANS ALL BITS ARE Ø
         BNE   SYSPAUDB                   N. GO ON
         MVC   WORKCHAR(Ø2),=C'NO'        Y. SAY NO HERE
SYSPAUDB MVC   ZPRMCL16+Ø8(48),WORKCHAR
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> CONDBAT  - MAX NO. CONNECTED DBAT
         MVC   ZPRMCL16(Ø8),=C'CONDBAT='
         LH    R9,SYSPCDB                 GET NUMBER OF CONNECTED DBATS
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø8(7),D           PACK TO ZONE
         OI    ZPRMCL16+14,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø8       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø8(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> CTHREAD  - MAX NO OF CONCURRENT THREADS
         MVC   ZPRMCL16(Ø8),=C'CTHREAD='
         LH    R9,SYSPCT                  GET CONCURRENT THD
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø8(7),D           PACK TO ZONE
         OI    ZPRMCL16+14,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø8       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø8(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> DLDFREQ  - CHECKPOINTS PER LEVEL ID UPDATE
         MVC   ZPRMCL16(Ø8),=C'DLDFREQ='
         LH    R9,SYSPDFRQ                GET CONCURRENT THD
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø8(7),D           PACK TO ZONE
         OI    ZPRMCL16+14,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø8       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø8(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> IDBACK   - MAX NO OF BACKGROUND IDS
         MVC   ZPRMCL16(Ø7),=C'IDBACK='
         LH    R9,SYSPIDB                 GET BACKGROUND IDS
         CVD   R9,D                       CONVERT DECIMAL
```

```
         UNPK  ZPRMCL16+Ø7(7),D           PACK TO ZONE
         OI    ZPRMCL16+13,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø7       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø7(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> IDFORE   - MAX NO OF FOREGROUND IDS
         MVC   ZPRMCL16(Ø7),=C'IDFORE='
         LH    R9,SYSPIDF                 GET FOREGROUND IDS
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø7(7),D           PACK TO ZONE
         OI    ZPRMCL16+13,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø7       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø7(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> LOGLOAD  - LOGLOAD VALUE CHECKPOINT FREQUENCY
         MVC   ZPRMCL16(Ø8),=C'LOGLOAD='
         SR    R9,R9                      ZERO REGISTER
         L     R9,SYSPLOGL                GET ZPARM VALUE
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø8(15),D          PACK TO ZONE
         OI    ZPRMCL16+22,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø8       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø8(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> MAXDBAT  - MAX NO OF ACTIVE REMOTE THREADS
         MVC   ZPRMCL16(Ø8),=C'MAXDBAT='
         LH    R9,SYSPRMT                 GET MAXIMUM ACTIVE REMOTE THD
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø8(7),D           PACK TO ZONE
         OI    ZPRMCL16+14,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø8       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø8(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> MON      - MONITOR TRACING FLAG
         MVC   ZPRMCL16(Ø4),=C'MON='
         MVC   WORKB32,SYSPMON            GET MONITOR TRACING FLAGS
         BAS   R14,BIT16RTN               CONVERT FIRST 16 BITS TO NUM
         BAS   R14,BIT32RTN               CONVERT NEXT 16 BITS TO NUM
```

```
         CLI   WORKCHR1,C')'              IF ALL BITS ARE Ø
         BNE   *+1Ø                       N. GO ON
         MVC   WORKCHAR(2),=C'NO'         Y. SAY NO
         MVC   ZPRMCL16+Ø4(48),WORKCHAR
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> MONSIZE  - MONITOR BUFFER SIZE
         MVC   ZPRMCL16(Ø8),=C'MONSIZE='
         SR    R9,R9                      ZERO REGISTER
         L     R9,SYSPMONS                GET MONITOR SIZE
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø8(15),D          PACK TO ZONE
         OI    ZPRMCL16+22,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø8       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø8(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> PCLOSEN  - NUMBER OF CHECKPOINT FOR READ ONLY SWITCHING
         MVC   ZPRMCL16(Ø8),=C'PCLOSEN='
         LH    R9,SYSPFRQ                 GET CONCURRENT THD
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø8(7),D           PACK TO ZONE
         OI    ZPRMCL16+14,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø8       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø8(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> PCLOSET  - MINUTES TO PSEUDO-CLOSE  READ ONLY SWITCHING
         MVC   ZPRMCL16(Ø8),=C'PCLOSET='
         LH    R9,SYSPTMR                 GET CONCURRENT THD
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø8(7),D           PACK TO ZONE
         OI    ZPRMCL16+14,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø8       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø8(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> RLF      - RESOURCE LIMIT FACILITY ENABLED
         MVC   ZPRMCL16(Ø8),=C'RLF=YES '
         TM    WRKPFLG1,B'ØØØØØØ1Ø'       BIT 7 ON
         BO    *+1Ø                       Y.
         MVC   ZPRMCL16+Ø4(Ø3),=C'NO '
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
```

```
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> RLFTBL   - RESOURCE LIMIT FACILITY TABLE ID
         MVC   ZPRMCL16(Ø7),=C'RLFTBL='
         MVC   ZPRMCL16+Ø7(L'SYSPRLFT),SYSPRLFT GET ZPARM VALUE
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> RLFERR   - RESOURCE LIMIT FACILITY ERROR
         MVC   ZPRMCL16(Ø7),=C'RLFERR='
         SR    R9,R9                      ZERO REGISTER
         L     R9,SYSPRLFR+1              GET LIMIT SU
         CVD   R9,D                       CONVERT DECIMAL
         UNPK  ZPRMCL16+Ø7(15),D          PACK TO ZONE
         OI    ZPRMCL16+21,X'FØ'          FIX LAST DIGIT
         MVC   ZEROHOLD,ZPRMCL16+Ø7       MOVE NUMBER IN HOLD AREA
         BAS   R14,DZERORTN               DROP LEADING ZEROS
         MVC   ZPRMCL16+Ø7(16),ZEROHOLD   MOVE TRUNCATED NUMBER BACK
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> RLFAUTH  - RESOURCE LIMIT FACILITY
         MVC   ZPRMCL16(Ø8),=C'RLFAUTH='
         MVC   ZPRMCL16+Ø8(Ø8),SYSPRLFA   GET ZPARM VALUE
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> ROUTCDE  - SYSTEM MESSAGE ROUTING CODE
         MVC   ZPRMCL16(Ø8),=C'ROUTCDE='
         MVC   WORKB16,SYSPSMRC           GET 32 BITS
         BAS   R14,BIT16RTN               CONVERT FIRST 16 BITS TO NUM
         CLI   WORKCHR1,C')'              IF ) MEANS ALL BITS ARE Ø
         BNE   SYSPROUB                   N. GO ON
         MVC   WORKCHAR(Ø2),=C'NO'        Y. SAY NO HERE
SYSPROUB MVC   ZPRMCL16+Ø8(48),WORKCHAR
         TRT   ZPRMCL16,TRTABLE           FIND FIRST BLANK
         MVI   Ø(1),C','                  PLUG COMMA HERE
         BAS   R14,ZWRTRTN             DO PRINT LINE
*> SMFACCT  - SMF ACCOUNTING FLAGS
         MVC   ZPRMCL16(Ø8),=C'SMFACCT='
         MVC   WORKB32,SYSPSMFA           GET SMF ACCOUNTING FLAGS
         BAS   R14,BIT16RTN               CONVERT FIRST 16 BITS TO NUM
         BAS   R14,BIT32RTN               CONVERT NEXT 16 BITS TO NUM
         CLI   WORKCHR1,C')'              IF ) MEANS ALL BITS ARE Ø
         BNE   SYSPACCB                   N. GO ON
         MVC   WORKCHAR(Ø2),=C'NO'        Y. SAY NO HERE
```

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

Rolf Loeben (Germany)

DataMirror has announced its Transformation Server for DB2/MVS, a source replication engine capable of sharing IBM System/390 data with nativeAS/400, Oracle, Sybase, Informix, and Microsoft SQL Server databases. It supports both DB2 for MVS/ESA and DB2 UDB for OS/390. The software is used for loading and replenishing data marts and data warehouses, sharing data with other corporate data stores, and distributing System/390-based data to Internet applications for mobile access, e-business, and Enterprise Information Portals.

For further information contact:
DataMirror Corporation, 3100 Steeles Avenue East, Suite 700, Markham, Ontario, Canada, L3R 8T3.
Tel: (905) 4150310.
DataMirror (UK), Windmill Court, Millfield Lane, Lower Kingswood, Tadworth, Surrey, KT20 6DL, UK.
Tel: (01737) 830770.
URL: http://www.datamirror.com.

* * *

IBM has announced DB2 Forms, for building application front ends to DB2 workstation databases. Applications can be created by developers, governed by administrators, and run by end users on Windows 95, 98, and NT 3.51 or later.

Extending DBEddit, it requires no database gateways, middleware, or ODBC drivers. Users can build advanced database techniques and commands without programming or SQL knowledge.

Compliant with the Open Group's Distributed Relational Database Architecture (DRDA), global connectivity between DB2 Forms applications and multiple DB2 database platforms is available through most network types, including Internet connections, dedicated dial-up lines, TCP/IP intranets, and closed SNA environments.

DRDA support allows access to multi-vendor databases like IMS, VSAM, Oracle, and Microsoft SQL Server via the multi-database gateway DB2 DataJoiner.

For further information contact your local IBM representative.

* * *

Data Junction has announced Data Junction for DB2, a Windows-based visual design tool for building and testing data transformations that work between DB2 and other data formats. Data Junction integrates disparate database systems and lets users migrate critical data between applications.

Data Junction has also announced DJEngine for DB2, a programmable, embeddable engine for executing conversions designed with Data Junction. It can be used for data warehouses, data marts, and various data migration and replication strategies.

For further information contact:
Data Junction, 2201 Northland Drive, Austin, TX 78756, USA.
Tel: (512) 459 1308.
URL: http://www.datajunction.com.