91 DB2 update

May 2000

In this issue

DB2 Update

Published by

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

North American office

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

Contributions

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

DB2 Update on-line

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

Editor

Trevor Eddolls

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 1997 issue, are available separately to subscribers for £22.50 ($33.50) each including postage.

Printed in England.

Image copy, DSNTIAUL copy, and disaster recovery of DB2 objects

It can be a time-consuming and error-prone process, preparing image copy jobs for tablespaces and DSNTIAUL copy jobs for tables that require GDGs to be defined, and writing JCL for each object created. With this procedure, only two SQL routines are needed to select all the tablespaces and tables to be backed up. DEFGALL REXX will prepare JCL that creates all undefined GDGs for all tablespaces and tables. ICOPYALL REXX will prepare JCL that takes image copies of all tablespaces and DSNTIAUL copies of all tables that are selected by the SQL. Finally, RECALL REXX will prepare disaster recovery jobs for all tablespaces and indexes.

The SQL to select tablespaces to be backed up is as follows:

```
SELECT DBNAME,NAME
FROM SYSIBM.SYSTABLESPACE A
WHERE NOT EXISTS (SELECT * FROM SYSIBM.SYSCOPY B
                        WHERE A.NAME=B.TSNAME AND
                        CURRENT TIMESTAMP - B.TIMESTAMP  < 50000
                    AND DEVTYPE='3590-1' AND DSNAME LIKE 'ODM%')
                AND DBNAME NOT LIKE 'WRK%'
           AND NAME NOT LIKE 'UNLOAD%'
           AND NAME NOT LIKE 'PCPY%'
           AND DBNAME NOT LIKE 'DSN%'
           AND DBNAME NOT LIKE 'DSQ%'
ORDER BY DBNAME,NAME;
```

The SQL to select tables to be backed up using the DSNTIAUL utility is as follows:

```
SELECT CREATOR,NAME
FROM SYSIBM.SYSTABLES
WHERE   SUBSTR(TSNAME,1,4) IN
('PABN','PALM','PHIS','PHST','PKAM','PDBA',
                                      'PKRE','PMKL','PPCK','PPLN','PVER')
AND
                  TSNAME NOT IN ('PABNP006','PALMP002') AND
                  TYPE='T'
ORDER BY 1,2 ;
```

These SQL routines are written in a member of a PDS, and only member names are passed to REXX EXECs. So, the same REXX can

be used for different SQL routines by passing different SQL member names.

PREPARING DEFINE GDG JCL (DEFGDG)

GDG names for tablespaces vary according to the partition parameter. If the partition parameter is 'YES', the GDG names for tablespaces will be as follows:

```
ODM.GB0.S1D.tablespace name.Ppartition number
```

If the partition parameter is 'NO', the GDG names are as follows:

```
ODM.GB0.S1D.tablespace name
```

Since table names can contain special characters that cannot be used as the first character of the dataset qualifier, and table names can be longer than eight characters, which a dataset qualifier cannot, a conversion is applied to all table names. The GDG names for tables can be obtained by using the following REXX.

TABCONV

```
/*   REXX   */
/*****************************************************************/
/*      THIS REXX CONVERTS TABLE NAMES TO GDG DATASET NAME       */
 SAY 'ENTER TABLE NAME ( DO NOT ENTER AUTH ID ) :'
 PULL TAB_NAME
 SAY 'ENTER DB2 SUBSYSTEM ID ( D,T,E,P ) :'
 PULL DB2ID
 PN=TRANSLATE(TAB_NAME,'X','_')
 IF LENGTH(PN) < 8 THEN ,
   UNL_DS_NAME=DB2ID||PN
 IF LENGTH(PN) > 7 & LENGTH(PN) < 15 THEN ,
   UNL_DS_NAME=DB2ID||SUBSTR(PN,1,7)||'.X'||,
               SUBSTR(PN,8,LENGTH(PN)-7)
 IF LENGTH(PN) > 14 THEN ,
   UNL_DS_NAME=DB2ID||SUBSTR(PN,1,7)||'.X'||,
               SUBSTR(PN,8,7)||'.X'||,
               SUBSTR(PN,15,LENGTH(PN)-14)
 DS_NAME = 'GDG DSN NAME = UNLOAD.GB0.'||UNL_DS_NAME
  SAY DS_NAME
```

For example, if the table name is NON_CST_PSTG_ENT and the subsystem-id is P, the GDG name for this table will be UNLOAD.GB0. PNONXCST.XXPSTGXE.XNT.

PREPARING IMAGE COPY AND DSNTIAUL COPY JCL (ICOPYALL)

ICOPYALL REXX can be run for DSNTIAUL copy, image copy, or both of them. Image and DSNTIAUL copy jobs are created in a dataset. Member names of the jobs are passed to the REXX. Retention period information for the copy datasets is also passed.

DSNTIAUL copy can be restarted from a given table. Since the query that selects tables gives the table names in alphabetical order, all tables coming before a given table name can be skipped. In order to restart from a given table, the 'RESTART' parameter must be coded and a table name must be entered for SYS2.OPERLIB(UNLRESTB) member.

The DSNTIAUL copy job consists of steps that each take a DSNTIAUL copy of a number of tables. ICOPYALL takes this number as a parameter. This parameter cannot be greater than 99 or less than 1.

An image copy job can be run at the partition level or tablespace level. If the JCL is run at the partition level, the define GDG JCL must also be run at the partition level. A parameter is passed to REXX to divide the job into parts. This parameter defines the number of jobs to be prepared. This option enables us to run image copy jobs on different tape units at the same time. Tablespaces are grouped in the jobs by their sizes, so that the run-time of the jobs will be approximately the same. The image copy restart mechanism is implemented in the tablespace select query. In the query, the tablespaces that have not taken an image copy for some time are selected. So, tablespaces that had an image copy taken in the previous run will not be selected.

PREPARING DISASTER RECOVERY JCL (RECALL)

For disaster recovery, two sets of JCL are created – tablespace recovery JCL and index recovery JCL. Tablespace recovery JCL can be prepared at the tablespace level or partition level. For each tablespace or partition, the last image copy taken in the local site and its volume serial number are found. For each volume serial number, a separate job will be prepared in order to run them at the same time. Recovery can be TOCOPY or END OF LOG recovery according to the parameter passed.

For index recovery, a parameter that indicates the number of jobs to

run is entered. However, for large indexes a new job is created. For these indexes, work datasets are catalogued in order to run the recovery index utility from the BUILD or SORT phase. For small indexes, the work datasets are not catalogued.

DEFGDG

```
/*   REXX   */
 PARSE ARG DB2ID DEFGDGMEM TSSELECT TBSELECT PARTITION
 £ALLOC FI(SYSPRINT) RECFM(F B) LRECL(133) SPACE(1,1) BLOCK(4Ø96)£
 £ALLOC FI(SYSPUNCH) RECFM(F B) LRECL(8Ø) SPACE(1,1) BLOCK(4Ø96)£
 £ALLOC FI(SYSIN) RECFM(F B) LRECL(8Ø) BLKSIZE(8Ø)£
 £ALLOC FI(DFJCL) DA('SYS2.BACKUP.JCLLIB(£||DEFGDGMEM||£)') SHR£
 £ALLOC FI(TSSEL) DA('SYS2.BACKUP.JCLLIB(£||TSSELECT||£)') SHR£
 £ALLOC FI(TBSEL) DA('SYS2.BACKUP.JCLLIB(£||TBSELECT||£)') SHR£
 CALL JOB_CARD;
 £EXECIO * DISKR TSSEL (STEM SEL.£
 £EXECIO * DISKR TSSEL (FINIS£
 £ALLOC FI(SYSRECØØ) SPACE(1,1) BLOCK(4Ø96)£
 CALL DO_SELECT;
 £EXECIO * DISKR SYSRECØØ (STEM TSNAMES.£
 £EXECIO * DISKR SYSRECØØ (FINIS£
 £FREE FI(SYSRECØØ)£
 DO MAIN_LP=1 TO TSNAMES.Ø
   TSNAME=STRIP(SUBSTR(TSNAMES.MAIN_LP,9,8),'B')
   IF PARTITION='YES' THEN DO;
     SEL.Ø=3;
     SEL.1=' SELECT CHAR(DECIMAL(PARTITIONS)) '
     SEL.2=' FROM SYSIBM.SYSTABLESPACE'
     SEL.3=' WHERE NAME='||''''||TSNAME''''||';'
     £ALLOC FI(SYSRECØØ) SPACE(1,1) BLOCK(4Ø96)£
     CALL DO_SELECT;
     £EXECIO * DISKR SYSRECØØ (STEM PARTCNT.£
     £EXECIO * DISKR SYSRECØØ (FINIS£
     £FREE FI(SYSRECØØ)£
     PARTCNT.1=SUBSTR(PARTCNT.1,2,5);
     PARTCNT.1=STRIP(TRANSLATE(PARTCNT.1,' ','ØØ'X),'B');
     DO WHILE ( SUBSTR(PARTCNT.1,1,1) = 'Ø' & LENGTH(PARTCNT.1) > 1 )
       IF SUBSTR(PARTCNT.1,1,1)='Ø' THEN ,
         PARTCNT.1=SUBSTR(PARTCNT.1,2,LENGTH(PARTCNT.1)-1)
     END;
     TSPARTCNT.MAIN_LP=PARTCNT.1
   END
   ELSE TSPARTCNT.MAIN_LP=Ø
   IF TSPARTCNT.MAIN_LP=Ø THEN DO
     GDG_NAME='ODM.GBØ.S1D'||'.'||TSNAME
     CALL DEFINE_GDG
   END
   ELSE DO
     DO III=1 TO TSPARTCNT.MAIN_LP
```

```
      GDG_NAME='ODM.GBØ.S1D'||'.'||TSNAME||'.P'||III
      CALL DEFINE_GDG
    END
  END
END
£EXECIO * DISKR TBSEL (STEM SEL.£
£ALLOC FI(SYSRECØØ) SPACE(1,1) BLOCK(4Ø96)£
CALL DO_SELECT
£EXECIO * DISKR SYSRECØØ (STEM TBNAMES.£
£EXECIO * DISKR SYSRECØØ (FINIS£
£FREE FI(SYSRECØØ)£
DO MAIN_LP=1 TO TBNAMES.Ø
  TBNAME=STRIP(SUBSTR(TBNAMES.MAIN_LP,1,8),'B') || '.' || ,
         STRIP(TRANSLATE(SUBSTR(TBNAMES.MAIN_LP,11,18),' ','ØØ'X),'B')
  PM=STRIP(TRANSLATE(SUBSTR(TBNAMES.MAIN_LP,11,18),' ','ØØ'X),'B')
  PN=TRANSLATE(PM,'X','_')
  IF LENGTH(PN) < 8 THEN ,
    UNL_DS_NAME=DB2ID||PN
  IF LENGTH(PN) > 7 & LENGTH(PN) < 15 THEN ,
    UNL_DS_NAME=DB2ID||SUBSTR(PN,1,7)||'.X'||,
                SUBSTR(PN,8,LENGTH(PN)-7)
  IF LENGTH(PN) > 14 THEN ,
    UNL_DS_NAME=DB2ID||SUBSTR(PN,1,7)||'.X'||,
                SUBSTR(PN,8,7)||'.X'||,
                SUBSTR(PN,15,LENGTH(PN)-14)
  GDG_NAME='UNLOAD.GBØ.'||UNL_DS_NAME
  CALL DEFINE_GDG
END
PUSH ''
PUSH '/*'
£EXECIO * DISKW DFJCL (FINIS£
STAT=MSG('OFF')
£FREE FI(SYSRECØØ)£
£FREE FI(SYSPRINT)£
£FREE FI(SYSPUNCH)£
£FREE FI(SYSIN)£
£FREE FI(TSSEL)£
£FREE FI(TBSEL)£
£FREE FI(DFJCL)£
STAT=MSG('ON')
£SUBMIT 'SYS2.BACKUP.JCLLIB(£DEFGDGMEM£)'£
EXIT
DEFINE_GDG:PROCEDURE EXPOSE EXPOSE GDG_NAME
  SAY GDG_NAME
  PUSH ''
  PUSH '  END'
  PUSH '    SET LASTCC=Ø'
  PUSH '  ELSE'
  PUSH '  END'
  PUSH '    IF LASTCC EQ Ø THEN SET MAXCC=Ø'
  PUSH '       LIMIT(8) NOEMPTY SCRATCH )'
  LINE='    DEFINE GDG ( NAME('||GDG_NAME||') -'
```

```
  PUSH LINE
  PUSH '  IF LASTCC GE 4 THEN DO'
  LINE='  LISTCAT ENT('||GDG_NAME||') GDG'
  PUSH LINE
  £EXECIO * DISKW DFJCL£
RETURN
JOB_CARD:PROCEDURE
  PUSH ''
  PUSH '//SYSIN    DD *'
  PUSH '//SYSPRINT DD SYSOUT=*'
  PUSH '//DEFINE EXEC PGM=IDCAMS'
  PUSH '//   MSGLEVEL=(1,1),REGION=4M'
  PUSH '//DEFGDG    JOB (ACCT#),'DEFINEGDG',MSGCLASS=X,CLASS=P,'
  £EXECIO * DISKW DFJCL£
RETURN
DO_SELECT:PROCEDURE EXPOSE SEL. DB2ID
  PUSH ''
  DO X = SEL.Ø TO 1 BY -1
    PUSH SEL.X
  END
  £EXECIO * DISKW SYSIN (FINIS£
  CMD = £RUN PROGRAM(DSNTIAUL) PLAN(DSNTIAUL)£
  CMD = CMD || ' LIB('''||DB2ID||'DSN.RUNLIB.LOAD''')'
  CMD = CMD || £ PARMS('SQL')£
  QUEUE 'END '
  IF DB2ID='D' THEN 'DSN SYSTEM(DBDØ)'
  IF DB2ID='T' THEN 'DSN SYSTEM(DBTØ)'
  IF DB2ID='E' THEN 'DSN SYSTEM(DBEØ)'
  IF DB2ID='P' THEN 'DSN SYSTEM(DBPØ)'
  IF RC > Ø THEN DO
    SAY 'CAN NOT CONNECT TO DB2 SUBSYSTEM.'
    SAY 'PLEASE TRY LATER...'
    RETURN
  END
  QUEUE CMD
  QUEUE 'END '
  IF DB2ID='D' THEN 'DSN SYSTEM(DBDØ)'
  IF DB2ID='T' THEN 'DSN SYSTEM(DBTØ)'
  IF DB2ID='E' THEN 'DSN SYSTEM(DBEØ)'
  IF DB2ID='P' THEN 'DSN SYSTEM(DBPØ)'
  £EXECIO * DISKR SYSPRINT (STEM SQLHATA.£
  UNLD_OK=Ø
  DO SQ_LP=1 TO SQLHATA.Ø
    IF INDEX(SQLHATA.SQ_LP,'DSNT495I SUCCESSFUL UNLOAD') > Ø THEN ,
      UNLD_OK=1
  END
  IF UNLD_OK=Ø THEN DO
    SAY 'THERE IS AN ERROR IN SQL STATEMENT.'
    DO SQ_LP2=1 TO SQLHATA.Ø
      SAY SQLHATA.SQ_LP2
    END
    EXIT 2Ø
```

```
   END
   PUSH ''
   £EXECIO * DISKW SYSPRINT (FINIS£
 RETURN
```

DEFGDGX

```
//DEFGDGX JOB (ACCT#),'',MSGCLASS=X,CLASS=P,
// MSGLEVEL=(1,1),REGION=4M
//*************************************************************
//* DEFGDGRX DB2ID MEMBER_NAME TS_SELECT TB_SELECT
//* DB2ID           : DB2 SUBSYSTEM ID. IT MAY BE D, T, E, OR P
//* MEMBER_NAME     : DEFINE GDG JCL NAME TO BE CREATED.
//* TS_SELECT       : QUERY THAT SELECTS TABLESPACES TO BE IMAGE
//*                 : COPIED.
//* TB_SELECT       : QUERY THAT SELECTS TABLES TO BE COPIED
//*                   USING DSNTIAUL.
//* PARTITION       : 'YES' IF RECOVERY IS MADE PARTITIONED
//*                     TABLESPACE LEVEL, OTHERWISE 'NO'.
//*************************************************************
/*
//RUNEXEC1 EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=8192K
//STEPLIB  DD DSN=ISP.SISPLOAD,DISP=SHR
//         DD DSN=PDSN.SDSNLOAD,DISP=SHR
//SYSEXEC  DD   DSN=SYSPDBA.REXXLIB,DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
 EXECUTIL SEARCHDD(YES)
 %DEFGDG  P UPDDFG TSSELUPD TBSELALL NO
/*
```

ICOPYALL

```
/*   REXX   */
 PARSE ARG DB2ID OP_ID UNLD_INTERVAL ICOPY_DSNAME UNLD_DSNAME ,
           RETPD_COPY RETPD_UNLD TSSELECT TBSELECT JC ,
           JOB_COUNT PARTITION IS_RESTART
 £ALLOC FI(SYSPRINT) RECFM(F B) LRECL(133) SPACE(1,1) BLOCK(4096)£
 £ALLOC FI(SYSPUNCH) RECFM(F B) LRECL(80) SPACE(1,1) BLOCK(4096)£
 £ALLOC FI(SYSIN) RECFM(F B) LRECL(80) BLKSIZE(80)£
 IF OP_ID='COPY' | OP_ID='BOTH' THEN DO;
   £ALLOC FI(ICJCL) DA('SYS2.BACKUP.JCLLIB(£||ICOPY_DSNAME||£)') SHR£
   £ALLOC FI(TSSEL) DA('SYS2.BACKUP.JCLLIB(£||TSSELECT||£)') SHR£
   JOB_NUM=0;
   CALL JOB_CARD_ICOPY;
   LABEL_NUM=0;
   /*  FIND TOTAL ACTIVE PAGE OF ALL TABLESPACES TO BE BACKED UP  */
   SEL.0=6;
   SEL.1='  SELECT CHAR(DECIMAL(SUM(NACTIVE)))'
   SEL.2='  FROM SYSIBM.SYSTABLESPACE'
```

```
SEL.3='  WHERE DBNAME NOT LIKE '||''''||'WRK%'||''''
SEL.4='  AND NAME NOT LIKE '||''''||'UNLOAD%'||''''
SEL.5='  AND NAME NOT LIKE '||''''||'PCPY%'||''''
SEL.6='  AND DBNAME NOT LIKE '||''''||'DSN%'||''''||';'
£ALLOC FI(SYSREC00) SPACE(1,1) BLOCK(4096)£
CALL DO_SELECT;
£EXECIO * DISKR SYSREC00 (STEM TOTPAG.£
£EXECIO * DISKR SYSREC00 (FINIS£
£FREE FI(SYSREC00)£
TOTPAG.1=SUBSTR(TOTPAG.1,2,11);
DO WHILE (SUBSTR(TOTPAG.1,1,1) = '0' & LENGTH(TOTPAG.1) > 1 )
  IF SUBSTR(TOTPAG.1,1,1)='0' THEN ,
    TOTPAG.1=SUBSTR(TOTPAG.1,2,LENGTH(TOTPAG.1)-1)
END;
SAY 'TOTAL PAGE NUMBER OF ALL TABLESPACES TO BE BACKED UP =' ,
                        TOTPAG.1
EVERY_JOB_HAS_ACTIVE_PAGE=TOTPAG.1/JOB_COUNT
EVERY_JOB_HAS_ACTIVE_PAGE=EVERY_JOB_HAS_ACTIVE_PAGE*1.1
/***   BEGIN OF TABLESPACE IMAGECOPY  ***/
£EXECIO * DISKR TSSEL (STEM SEL.£
£EXECIO * DISKR TSSEL (FINIS£
£ALLOC FI(SYSREC00) SPACE(1,1) BLOCK(4096)£
CALL DO_SELECT;
£EXECIO * DISKR SYSREC00 (STEM TSNAMES.£
£EXECIO * DISKR SYSREC00 (FINIS£
£FREE FI(SYSREC00)£
TOT_NACTIVE=0;
UPPERL=0;
DO MAIN_LP=1 TO TSNAMES.0
  TSNAME=STRIP(SUBSTR(TSNAMES.MAIN_LP,9,8),'B')
  SEL.0=3;
  SEL.1=' SELECT CHAR(DECIMAL(PARTITIONS)),CHAR(DECIMAL(NACTIVE))'
  SEL.2=' FROM SYSIBM.SYSTABLESPACE '
  SEL.3=' WHERE NAME='||''''||TSNAME''''||';'
  £ALLOC FI(SYSREC00) SPACE(1,1) BLOCK(4096)£
  CALL DO_SELECT;
  £EXECIO * DISKR SYSREC00 (STEM SQLOUT.£
  £EXECIO * DISKR SYSREC00 (FINIS£
  £FREE FI(SYSREC00)£
  PARTCNT.1=SUBSTR(SQLOUT.1,2,5);
  NACTIVE=SUBSTR(SQLOUT.1,9,11);
  PARTCNT.1=STRIP(TRANSLATE(PARTCNT.1,' ','00'X),'B');
  DO WHILE (SUBSTR(NACTIVE,1,1) = '0' & LENGTH(NACTIVE) > 1 )
    IF SUBSTR(NACTIVE,1,1)='0' THEN ,
      NACTIVE=SUBSTR(NACTIVE,2,LENGTH(NACTIVE)-1)
  END;
  TOT_NACTIVE=TOT_NACTIVE+NACTIVE
  SAY MAIN_LP 'TS NAME=' TSNAME 'NACTIVE=' NACTIVE
  DO WHILE ( SUBSTR(PARTCNT.1,1,1) = '0' & LENGTH(PARTCNT.1) > 1 )
    IF SUBSTR(PARTCNT.1,1,1)='0' THEN ,
      PARTCNT.1=SUBSTR(PARTCNT.1,2,LENGTH(PARTCNT.1)-1)
  END;
```

```
    TSPARTCNT.MAIN_LP=PARTCNT.1
    /* IF IMAGE COPIES ARE TO BE TAKEN PARTITIONED TABLESPACE LEVEL,
       PARTITION VARIABLE WILL BE 'YES'. */
    IF PARTITION='NO' THEN TSPARTCNT.MAIN_LP=Ø;

    IF TOT_NACTIVE > EVERY_JOB_HAS_ACTIVE_PAGE THEN DO
      LABEL_NUM=Ø;
      LOWERL=UPPERL+1
      UPPERL=MAIN_LP-1
      CALL WRITE_SYSIN
      OPR_TYPE='COPY'
      CALL ADD_WTO
      CALL JOB_CARD_ICOPY
      TOT_NACTIVE=NACTIVE
    END;
    IF TSPARTCNT.MAIN_LP=Ø THEN PARTNUM_LAST=1
       ELSE PARTNUM_LAST=TSPARTCNT.MAIN_LP
    DO PARTNUM=1 TO PARTNUM_LAST
      CALL ADD_DD_COPY
    END;
  END
  PUSH ''
  PUSH '/*'
  LOWERL=UPPERL+1
  UPPERL=MAIN_LP-1
  CALL WRITE_SYSIN
  OPR_TYPE='COPY'
  CALL ADD_WTO;
  CALL MODIFY_JOB
  £FREE FI(TSSEL)£
  £FREE FI(ICJCL)£
END /* IF OP_ID='COPY' | OP_ID='BOTH' THEN DO; */
/***   END OF TABLESPACE IMAGECOPY  ***/
/***   BEGIN OF DSNTIAUL COPY   ***/
IF OP_ID='UNLD' | OP_ID='BOTH' THEN DO;
  £ALLOC FI(TBSEL) DA('SYS2.BACKUP.JCLLIB(£||TBSELECT||£)') SHR£
  £ALLOC FI(UNJCL) DA('SYS2.BACKUP.JCLLIB(£||UNLD_DSNAME|| ,
    £)') SHR£
  UN_IN=Ø
  UN_IN=UNLD_INTERVAL
  IF UN_IN > 99 | UN_IN <1 THEN DO
    SAY 'UNLOAD INTERVAL CAN NOT BE GREATER THAN 99 OR LESS THAT 1...'
    EXIT 2Ø
  END
  IF IS_RESTART='RESTART' THEN DO
    £EXECIO * DISKR SYSRESIN (STEM RESTART_TABLE.£
    FROM_THIS_TABLE=STRIP(RESTART_TABLE.1,'B')
    SAY 'PROCESS WILL START FROM ' FROM_THIS_TABLE
  END
  £EXECIO * DISKR TBSEL (STEM SEL.£
  £EXECIO * DISKR TBSEL (FINIS£
```

```
£ALLOC FI(SYSREC00) SPACE(1,1) BLOCK(4096)£
CALL DO_SELECT;
£EXECIO * DISKR SYSREC00 (STEM TBNAMES.£
£EXECIO * DISKR SYSREC00 (FINIS£
£FREE FI(SYSREC00)£
IF IS_RESTART='RESTART' THEN DO
  NEW_NUM=0
  DO LP_VAR10= 1 TO TBNAMES.0
    TBNAME=STRIP(SUBSTR(TBNAMES.LP_VAR10,1,8),'B') || '.' || ,
    STRIP(TRANSLATE(SUBSTR(TBNAMES.LP_VAR10,11,18),' ','00'X),'B')
    IF TBNAME <= FROM_THIS_TABLE THEN ITERATE
    NEW_NUM=NEW_NUM+1
    TBNAMES.NEW_NUM=TBNAMES.LP_VAR10
  END
  TBNAMES.0=NEW_NUM
END
INT_PART=0
MAIN_LP=0
LAST_DD_NAME=''
CALL JOB_CARD_UNLD;
CALL NEW_STEP
DO MAIN_LP=1 TO TBNAMES.0
  TBNAME=STRIP(SUBSTR(TBNAMES.MAIN_LP,1,8),'B') || '.' || ,
    STRIP(TRANSLATE(SUBSTR(TBNAMES.MAIN_LP,11,18),' ','00'X),'B')
  SAY MAIN_LP TBNAME
  INT_PART=MAIN_LP % UN_IN
  REMAINDER = MAIN_LP - INT_PART * UN_IN
  IF REMAINDER=1 & INT_PART <> 0 THEN DO
    FROM_TAB=INT_PART * UN_IN
    TO_TAB=(INT_PART - 1) * UN_IN + 1
    CALL UNLOAD_SYSIN
    CALL NEW_STEP
  END
  PM=STRIP(TRANSLATE(SUBSTR(TBNAMES.MAIN_LP,11,18),' ','00'X),'B')
  PN=TRANSLATE(PM,'X','_')
  IF LENGTH(PN) < 8 THEN ,
    UNL_DS_NAME=DB2ID||PN
  IF LENGTH(PN) > 7 & LENGTH(PN) < 15 THEN ,
    UNL_DS_NAME=DB2ID||SUBSTR(PN,1,7)||'.X'||,
                SUBSTR(PN,8,LENGTH(PN)-7)
  IF LENGTH(PN) > 14 THEN ,
    UNL_DS_NAME=DB2ID||SUBSTR(PN,1,7)||'.X'||,
                SUBSTR(PN,8,7)||'.X'||,
                SUBSTR(PN,15,LENGTH(PN)-14)
  CALL ADD_DD_UNLD;
END
IF TBNAMES.0 > 0 THEN DO
  FROM_TAB=TBNAMES.0
  TO_TAB=INT_PART  * UN_IN + 1
  IF REMAINDER=0 THEN TO_TAB=(INT_PART - 1)  * UN_IN + 1
  CALL UNLOAD_SYSIN
```

```
    OPR_TYPE='UNLOAD'
    CALL ADD_WTO;
    PUSH ''
    PUSH '//'
    £EXECIO * DISKW UNJCL(FINIS£
    £FREE FI(UNJCL)£
  END
  £FREE FI(TBSEL)£
END /* IF OP_ID='UNLD' | OP_ID='BOTH' THEN DO; */
£FREE FI(SYSPRINT)£
£FREE FI(SYSPUNCH)£
£FREE FI(SYSIN)£
IF OP_ID = 'UNLD' | OP_ID='BOTH' THEN ,
   £SUBMIT 'SYS2.BACKUP.JCLLIB(£UNLD_DSNAME£)'£
IF OP_ID = 'COPY' | OP_ID = 'BOTH' THEN ,
   £SUBMIT 'SYS2.BACKUP.JCLLIB(£ICOPY_DSNAME£)'£
EXIT
MODIFY_JOB:PROCEDURE EXPOSE TSNAMES. JC DB2ID
  CALL MODIFY_JOB_CARD
  PUSH ''
  PUSH '/*'
  DO MAIN_LP=TSNAMES.Ø TO 1 BY -1
    TSNAME=STRIP(SUBSTR(TSNAMES.MAIN_LP,1,8),'B') || '.' || ,
           STRIP(SUBSTR(TSNAMES.MAIN_LP,9,8),'B')
    LINE = '   MODIFY RECOVERY TABLESPACE ' || TSNAME || ,
           ' DELETE AGE(31)'
    PUSH LINE
  END
  PUSH '//DSNUPROC.SYSIN DD *'
  £EXECIO * DISKW ICJCL£
  PUSH ''
  PUSH '//'
  £EXECIO * DISKW ICJCL (FINIS£
RETURN;
MODIFY_JOB_CARD:PROCEDURE EXPOSE DB2ID JC DB2ID
  PUSH ''
  LINE='//MODIFY EXEC DSNUPROC,SYSTEM=DB'||DB2ID||'Ø' ,
            ',UID=''MODIFY'',UTPROC=''''
  PUSH LINE
  PUSH '// MSGLEVEL=(1,1),REGION=ØM,NOTIFY=SKMXSYP,TYPRUN=HOLD'
  LINE='//MODIFY  JOB (ACCT#),'MODIFY','||,
       'MSGCLASS=X,CLASS='||JC||','
  PUSH LINE
  £EXECIO * DISKW ICJCL£
RETURN
```

Editor's note: this article will be continued in the next issue.

Abdullah Ongul
DBA
Disbank (Turkey)

DSN1COPY generator utility – part 3

This month we conclude the REXX procedure, DCU, which generates several DSN1COPY JCL streams.

- DSN1COP2 – JCL skeleton:

```
)TBA 72
)CM -------------------------------------------------------------------
)CM Skeleton to generate DSN1COPY utility                              --
)CM -------------------------------------------------------------------
//&user.X JOB (1200-1205-00),'&option',
//             NOTIFY=&user,REGION=4M,
//             CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//* ******************************************************************
//*    &title
//*    GENERATION DATE AND TIME : &date AT: &time
//*    CALCULATING TIME IS &ctime SECONDS.
//*    DSN1COPY  - WAS RUN WITH THE FOLLOWING PARAMETERS:
//*    PARAMETER   PARAMETER VALUE
//*    ----------  ----------------------
//*    SSID     :  &db2
//*    Creator  :  &creC
//*    Name     :  &tabc
//*    Tsname   :  &tsnc
//*    Dbname   :  &dbnc
//*    Stopts   :  &sts
//*    Devt     :  &devt
//*    Retpd    :  &rpd
//*    Withindx :  &wix
//*  NUM  DATABASE  TABLESPACE   TRACKS   PART
//*  ---  --------  ----------  -------   ----
)DOT "ALIST"
//* &detail
)ENDDOT
//*  ---  --------  ----------  -------   ----
//*                       TOTAL:&tot TRACKS OR
//*                             &cyl CYLINDERS
//*  NAMING CONVENTION USED WITH DSN1COPY DATASETS:
//*     PART 1=&user..DCU
//*          2=DSN1CXXX, WHERE XXX = PARTITION NUMBER
//*          3=DBNAME  , WHERE DBNAME = DATABASE NAME
//*          4=TSNAME  , WHERE TSNAME = TABLESPACE NAME
//*-------------------------------------------------
//*---- DSN1COPY - TABLESPACES/INDEXSPACES
)SEL &sts = YES
//*---- STOP TABLESPACES/INDEXSPACES ------------------
//STOPTS  EXEC PGM=IKJEFT01,COND=(4,LT)
```

```
//STEPLIB  DD DSN=DSN51Ø.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
)DOT "ALIST"
)SEL &pr = Ø
   -STOP DATABASE(&db) SPACENAM(&ts)
)ENDSEL
)SEL &pr > Ø
   -STOP DATABASE(&db) SPACENAM(&ts) PART(&pr)
)ENDSEL
)ENDDOT
//*
)ENDSEL
)DOT "ALIST"
//*-------------------------------------------------
)SEL &pr = Ø
//*  DSN1COPY - OF &db..&ts
)ENDSEL
)SEL &pr > Ø
//*  DSN1COPY - OF &db..&ts PART &pr
)ENDSEL
//*-------------------------------------------------
//COPY&scu EXEC PGM=DSN1COPY,COND=(4,LT)
//STEPLIB  DD DSN=DSN51Ø.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
)SEL &devt = 339Ø
//SYSUT1   DD DISP=SHR,
//         DSN=&user..DCU.DSN1C&pr1..&db..&ts
)ENDSEL
)SEL &devt = TAPE
//SYSUT1   DD DISP=(,KEEP),
//         UNIT=TAPE,
//         LABEL=(&scu,RETPD=14),
)SEL &scu = 1
//         VOL=(PRIVATE,RETAIN),
)ENDSEL
)SEL &scu > 1
//         VOL=(PRIVATE,RETAIN,REF=*.COPY1.SYSUT2),
)ENDSEL
//         DSN=&user..DCU.DSN1C&pr1..&db..&ts
)ENDSEL
//SYSUT2   DD DSN=&catn..DSNDBD.&db..&ts..IØØØ1.A&pr1,
//         DISP=OLD
)ENDDOT
)SEL &sts = YES
//*---- START TABLESPACES/INDEXSPACES -----------------
//STARTS  EXEC PGM=IKJEFTØ1,COND=(4,LT)
//STEPLIB  DD DSN=DSN51Ø.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
```

```
   DSN SYSTEM(&db2)
)DOT "ALIST"
)SEL &pr = Ø
   -START DATABASE(&db) SPACENAM(&ts)
)ENDSEL
)SEL &pr > Ø
   -START DATABASE(&db) SPACENAM(&ts) PART(&pr)
)ENDSEL
)ENDDOT
   -DIS   DATABASE(*) SPACENAM(*) RESTRICT
//*
)ENDSEL
```

- DSN1COP3 – JCL skeleton:

```
)TBA 72
)CM ------------------------------------------------------------------
)CM Skeleton to generate DSN1COPY utility                             --
)CM ------------------------------------------------------------------
//&user.X JOB (12ØØ-12Ø5-ØØ),'&option',
//             NOTIFY=&user,REGION=4M,
//             CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//*    &title
//*    GENERATION DATE AND TIME : &date AT: &time
//*    DSN1COPY  - WAS RUN WITH THE FOLLOWING PARAMETERS:
//*    PARAMETER   PARAMETER VALUE
//*    ----------  ----------------------
//*    SSID     :  &db2
//*    Tosystem :  &sysi
//*    Location :  &loc
//*    Creator  :  &crec
//*    Name     :  &tabc
//*    Tsname   :  &tsnc
//*    Dbname   :  &dbnc
//*    Stopts   :  &sts
//*    Withindx :  &wix
//*    Runstat  :  &run
//*      ----------&db2----------    ----------&sysi----------
//*  NUM DATABASE TABLESPACE PART    DATABASE TABLESPACE PART
//*  --- -------- ---------- ----    -------- ---------- ----
)DOT "ALIST"
//* &detail
)ENDDOT
)SEL &wix = YES
//* ------------------------------------------------------------------
//*  NUM DATABASE INDEXSPACE PART    DATABASE INDEXSPACE PART
//*  --- -------- ---------- ----    -------- ---------- ----
)DOT "ILIST"
//* &line
)ENDDOT
)ENDSEL
```

```
//*-------------------------------------------------
//*  NAMING CONVENTION USED WITH DSN1COPY DATASETS:
//*     PART 1=&user..DCU
//*          2=DSN1CXXX, WHERE XXX = PARTITION NUMBER
//*          3=DBNAME  , WHERE DBNAME = DATABASE NAME
//*          4=TSNAME  , WHERE TSNAME = TABLESPACE NAME
//*-------------------------------------------------
//*---- DSN1COPY - TABLESPACES/INDEXSPACES
)SEL &sts = YES
//*---- STOP TABLESPACES &db2 ------------------------
//STOPTS  EXEC PGM=IKJEFT01,COND=(4,LT)
//STEPLIB  DD DSN=DSN510.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
)SET dbp = &z
)SET tsp = &z
)DOT "ALIST"
)SEL &dbp NE &db1 OR &tsp NE &ts1 OR &pr1 GT 0
)SEL &pr1 = 0
   -STOP DATABASE(&db1) SPACENAM(&ts1)
)ENDSEL
)SEL &pr1 > 0
   -STOP DATABASE(&db1) SPACENAM(&ts1) PART(&pr1)
)ENDSEL
)ENDSEL
)SET dbp = &db1
)SET tsp = &ts1
)ENDDOT
)SEL &wix = YES
)DOT "ILIST"
)SEL &ipr1 = 0
   -STOP DATABASE(&idb1) SPACENAM(&isp1)
)ENDSEL
)SEL &ipr1 > 0
   -STOP DATABASE(&idb1) SPACENAM(&isp1) PART(&ipr1)
)ENDSEL
)ENDDOT
)ENDSEL
//*---- STOP TABLESPACES &sysi ------------------------
//STOPTS  EXEC PGM=IKJEFT01,COND=(4,LT)
//STEPLIB  DD DSN=DSN610.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&sysi)
)SET dbp = &z
)SET tsp = &z
)DOT "ALIST"
)SEL &dbp NE &db3 OR &tsp NE &ts2 OR &pr1 GT 0
)SEL &pr1 = 0
   -STOP DATABASE(&db3) SPACENAM(&ts2)
```

```
)ENDSEL
)SEL &pr1 > Ø
   -STOP DATABASE(&db3) SPACENAM(&ts2) PART(&pr1)
)ENDSEL
)ENDSEL
)SET dbp = &db3
)SET tsp = &ts2
)ENDDOT
)SEL &wix = YES
)DOT "ILIST"
)SEL &ipr1 = Ø
   -STOP DATABASE(&idb2) SPACENAM(&isp2)
)ENDSEL
)SEL &ipr1 > Ø
   -STOP DATABASE(&idb2) SPACENAM(&isp2) PART(&ipr1)
)ENDSEL
)ENDDOT
)ENDSEL
//*
)ENDSEL
)SET dbp = &z
)SET tsp = &z
)DOT "ALIST"
)SEL &dbp NE &db1 OR &tsp NE &ts1 OR &pr1 GT Ø
//*-------------------------------------------------
)SEL &pr1 = Ø
//*  DSN1COPY - OF &db1..&ts1
)ENDSEL
)SEL &pr1 > Ø
//*  DSN1COPY - OF &db1..&ts1 PART: &pr1
)ENDSEL
//*-------------------------------------------------
//COPY&scu EXEC PGM=DSN1COPY,COND=(4,LT),
)SEL &prts = Ø
//    PARM='SEGMENT,OBIDXLAT,RESET'
)ENDSEL
)SEL &prts > Ø
//    PARM='OBIDXLAT,RESET,NUMPARTS(&prts)'
)ENDSEL
//STEPLIB  DD DISP=SHR,DSN=DSN61Ø.SDSNLOAD
)SEL &pr1 = Ø
)SET pr1 = ØØ1
//SYSUT1   DD DISP=OLD,DSN=&ca1..DSNDBD.&db1..&ts1..IØØØ1.A&pr1
//SYSUT2   DD DISP=OLD,DSN=&ca2..DSNDBD.&db3..&ts2..IØØØ1.A&pr1
)SET pr1 = ØØØ
)ENDSEL
)SEL &pr1 > Ø
//SYSUT1   DD DISP=OLD,DSN=&ca1..DSNDBD.&db1..&ts1..IØØØ1.A&pr1
//SYSUT2   DD DISP=OLD,DSN=&ca2..DSNDBD.&db3..&ts2..IØØØ1.A&pr1
)ENDSEL
//SYSPRINT DD SYSOUT=*
```

```
//SYSXLAT  DD *
  &xid1                  DBID FOR &db1
  &xid2                  PSID FOR &ts1
)ENDSEL
  &xid3                  OBID FOR &tab
)SET dbp = &db1
)SET tsp = &ts1
)ENDDOT
)SEL &wix = YES
)DOT "ILIST"
//*-----------------------------------------------------
)SEL &ipr1 = Ø
//*  DSN1COPY - OF &idb1..&isp1
)ENDSEL
)SEL &ipr1 > Ø
//*  DSN1COPY - OF &idb1..&isp1 PART: &ipr1
)ENDSEL
//*-----------------------------------------------------
//COPI&icu EXEC PGM=DSN1COPY,COND=(4,LT),
)SEL &ipr = Ø
//    PARM='SEGMENT,OBIDXLAT,RESET'
)ENDSEL
)SEL &ipr > Ø
//    PARM='OBIDXLAT,RESET,NUMPARTS(&ipr)'
)ENDSEL
//STEPLIB  DD DISP=SHR,DSN=DSN61Ø.SDSNLOAD
)SEL &ipr1 = Ø
)SET ipr1 = ØØ1
//SYSUT1   DD DISP=OLD,DSN=&ica1..DSNDBD.&idb1..&isp1..IØØØ1.A&ipr1
//SYSUT2   DD DISP=OLD,DSN=&ica2..DSNDBD.&idb2..&isp2..IØØØ1.A&ipr1
)SET ipr1 = ØØØ
)ENDSEL
)SEL &ipr1 > Ø
//SYSUT1   DD DISP=OLD,DSN=&ica1..DSNDBD.&idb1..&isp1..IØØØ1.A&ipr1
//SYSUT2   DD DISP=OLD,DSN=&ica2..DSNDBD.&idb2..&isp2..IØØØ1.A&ipr1
)ENDSEL
//SYSPRINT DD SYSOUT=*
//SYSXLAT  DD *
  &ixid1                  DBID   FOR &idb1
  &ixid2                  ISOBID FOR &isp1
  &ixid3                  OBID
)ENDDOT
)ENDSEL
)SEL &sts = YES
//*---- START TABLESPACES &db2 ------------------------
//STARTS  EXEC PGM=IKJEFTØ1,COND=(4,LT)
//STEPLIB  DD DSN=DSN51Ø.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
)SET dbp = &z
```

```
)SET tsp = &z
)DOT "ALIST"
)SEL &dbp NE &db1 OR &tsp NE &ts1 OR &pr1 GT Ø
)SEL &pr1 = Ø
   -START DATABASE(&db1) SPACENAM(&ts1)
)ENDSEL
)SEL &pr1 > Ø
   -START DATABASE(&db1) SPACENAM(&ts1) PART(&pr1)
)ENDSEL
)ENDSEL
)SET dbp = &db1
)SET tsp = &ts1
)ENDDOT
)SEL &wix = YES
)DOT "ILIST"
)SEL &ipr1 = Ø
   -START DATABASE(&idb1) SPACENAM(&isp1)
)ENDSEL
)SEL &ipr1 > Ø
   -START DATABASE(&idb1) SPACENAM(&isp1) PART(&ipr1)
)ENDSEL
)ENDDOT
)ENDSEL
   -DIS   DATABASE(*) SPACENAM(*) RESTRICT
//*
//*---- START TABLESPACES &sysi -----------------------
//STARTS  EXEC PGM=IKJEFTØ1,COND=(4,LT)
//STEPLIB  DD DSN=DSN61Ø.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&sysi)
)SET dbp = &z
)SET tsp = &z
)DOT "ALIST"
)SEL &dbp NE &db3 OR &tsp NE &ts2 OR &pr1 GT Ø
)SEL &pr1 = Ø
   -START DATABASE(&db3) SPACENAM(&ts2)
)ENDSEL
)SEL &pr1 > Ø
   -START DATABASE(&db3) SPACENAM(&ts2) PART(&pr1)
)ENDSEL
)ENDSEL
)SET dbp = &db3
)SET tsp = &ts2
)ENDDOT
)SEL &wix = YES
)DOT "ILIST"
)SEL &ipr1 = Ø
   -START DATABASE(&idb2) SPACENAM(&isp2)
)ENDSEL
)SEL &ipr1 > Ø
```

```
   -START DATABASE(&idb2) SPACENAM(&isp2) PART(&ipr1)
)ENDSEL
)ENDDOT
)ENDSEL
   -DIS   DATABASE(*) SPACENAM(*) RESTRICT
//*
)ENDSEL
)SEL &rus = YES
//*---- RUNSTATS          -------------------------
//RUNSTP EXEC DSNTPROC,SYSTEM=&sysi,COND=(4,LT),
//      UID='&user..RUNSTAT',UTPROC=''
//STEPLIB  DD DSN=DSN610.SDSNLOAD,DISP=SHR
//SYSIN    DD  *
)SET dbp = &z
)SET tsp = &z
)DOT "ALIST"
)SEL &dbp NE &db3 OR &tsp NE &ts2 OR &pr1 GT 0
)BLANK 1
)SEL &ipr1 = 0
  RUNSTATS TABLESPACE &db3..&ts2
)ENDSEL
)SEL &ipr1 > 0
  RUNSTATS TABLESPACE &db3..&ts2 PART(&ipr1)
)ENDSEL
               TABLE (ALL)
               INDEX (ALL)
             SHRLEVEL REFERENCE
               REPORT NO
               UPDATE ALL
)ENDSEL
)SET dbp = &db3
)SET tsp = &ts2
)ENDDOT
/*
)ENDSEL
```

- DSN1COP4 – JCL skeleton:

```
)TBA 72
)CM ---------------------------------------------------------------------
)CM Skeleton to generate DSN1COPY utility                                --
)CM ---------------------------------------------------------------------
//&user.X JOB (1200-1205-00),'&option',
//             NOTIFY=&user,REGION=4M,
//             CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//*    &title
//*    GENERATION DATE AND TIME : &date AT: &time
//*    CALCULATING TIME IS &ctime SECONDS.
//*    DSN1COPY  - WAS RUN WITH THE FOLLOWING PARAMETERS:
//*    PARAMETER   PARAMETER VALUE
```

```
//*    ----------  ----------------------
//*    SSID     :  &db2
//*    Creator  :  &creC
//*    Name     :  &tabc
//*    Stopts   :  &sts
//*-------------------------------------------------
//*    Icdate   : &icd
//*    Ictime   : &ict
//*    Vcatname : &vc
//*    Dbname   : &db
//*    Tsname   : &ts
//*    Dsname   : &dsn
//*-------------------------------------------------
//*---- DSN1COPY - TABLESPACE
)SEL &sts = YES
//*---- STOP TABLESPACES -----------------------------
//STOPTS  EXEC PGM=IKJEFT01,COND=(4,LT)
//STEPLIB  DD DSN=DSN510.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
   -STOP DATABASE(&db) SPACENAM(&ts)
)ENDSEL
//*-------------------------------------------------
//COPYIC   EXEC PGM=DSN1COPY,PARM='FULLCOPY',COND=(4,LT)
//STEPLIB  DD DSN=DSN510.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=SHR,
//         DSN=&dsn
//SYSUT2   DD DISP=OLD,
//         DSN=&vc..DSNDBD.&db..&ts..I0001.A001
)SEL &sts = YES
//*---- START TABLESPACES ----------------------------
//STARTS  EXEC PGM=IKJEFT01,COND=(4,LT)
//STEPLIB  DD DSN=DSN510.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
   -START DATABASE(&db) SPACENAM(&ts)
//*
)ENDSEL
```

- DSNC00 – message:

```
DSNC001       .ALARM = YES  .WINDOW=NORESP .ALARM = YES
'&message'
```

Bernard Zver
Database Administrator
Informatika Maribor (Slovenia)

DB2 DDL syntax checker for the CREATE and ALTER statement

A DBA is always struggling with standards and technical recommendations. This is often the case when dealing with DB2 Data Definition Language (DDL) statements, regardless of their source (manually created, provided by software vendors, or generated by database modelling tools).

There are many reasons why DBAs must edit DB2 DDL statements, the most common one being to retrofit the DDL statements to their installation's standards. Manual editing of DB2 DDL is a time-consuming error-prone activity. Some of the simpler changes can be accomplished with a few edit commands. In other cases, such as when you want to specify different parameters for the same clause (for example, tablespaces to use bufferpool BP1 and index spaces to use bufferpool BP5), a more complex and time-consuming edit process is required. Whenever computations are required, such as in the case of PRIQTY, SECQTY, and SEGSIZE, the editing process becomes a very tedious chore, one that may not be completed in the required timeframe.

The CREATE DATABASE, CREATE TABLESPACE, and CREATE INDEX statements have many clauses, each with different types of parameter, and, in many cases, the clauses are correlated to other clauses (such as in the case of index type and subpage clauses). All of these options make manual editing and validating a difficult and lengthy process.

There are many products that perform syntax checks of DB2 DDL statements. If you have any of these products, I strongly recommend that you become familiar with them. It will save you lots of time and headaches. However, be aware that some of these products have limitations. Some of the limitations I have found in DB2 DDL syntax checkers are as follows:

- The DDL syntax checker product may not take into consideration what version of DB2 the DDL is for, thus not verifying whether the clauses and/or parameters are appropriate or not.

- The DDL syntax checker product may not take into consideration the DASD device geometry to be used when reviewing and/or computing PRIQTY, SECQTY, and SEGSIZE parameters.
- Most of the DDL syntax checker products I have worked with *do not* allow you to specify your installation standards and retrofit (edit) them into the DDL being checked.

The purpose of this article is to introduce you to a DBA tool, an ISPF application that will allow you to:

- Perform a syntax check on the ALTER and CREATE statements for database, tablespace, and index objects; the syntax check will take into consideration the DB2 version specified by the user to determine which clauses and parameters are valid. Correlation between different clauses will also be taken into consideration. Error or information messages will be inserted into the DDL.
- Perform all computations for PRIQTY, SECQTY, and SEGSIZE based on the device geometry and update the DDL with these new values. Optionally update the DDL to enforce your installation standards for bufferpool names, storage, and/or vcat group names and index type. Optionally insert comments describing the DDL differences from the user-specified standards.

The DBA syntax checker tool is an ISPF dialog (application) by the name of CDB2DDL (for Check DB2 DDL). It is composed of an ISPF edit macro written using the REXX language, six ISPF panels, and one set of ISPF messages.

There are two options for installing this ISPF application. Both options require that you copy the CDB2DDL edit macro into one of the libraries allocated under the ISPCLIB or ISPEXEC DDname of your TSO JCL log-on procedure. Option 1 requires that the six panels (PDB2DDL1, PDB2DDL2, PDB2DDL3, PDB2DDLC, PDB2DDLH, and PDB2DDLI) be copied into one of the libraries allocated under the ISPPLIB DDname of your TSO JCL log-on procedure. Also, the MDDL00 member containing the panel's messages must be copied into one of the libraries allocated under the ISPMLIB DDname of your TSO JCL log-on procedure. Option 2 requires that you edit the CDB2DDL edit macro: uncomment and modify the two LIBDEF

statements found in lines 45 and 46. The LIBDEF statements must point to the PDS that contains the panels and messages members.

This DBA tool will run in any MVS or OS/390 environment that has TSO (with REXX support) and ISPF installed. It will also run in any personal computer that has the SPF/PC 4.0 product by CTC installed.

DB2 Update, Issue 72, October 1998, contained an article entitled *Peculiarities of the CREATE statement clauses*, which explains the importance of taking into consideration the DASD device geometry when specifying the PRIQTY, SECQTY, and SEGSIZE parameters. Most of the article's concepts have been incorporated into the CDB2DDL edit macro. The most important concepts incorporated into the CDB2DDL edit macro are related to the computation of the PRIQTY and SECQTY values. These concepts are as follows:

- The PRIQTY and SECQTY values will be affected by the DASD device geometry.
- The quantity values will never be higher than the maximum USABLE size allowed by DB2.
- For segmented tablespaces, the SEGSIZE parameter value will be in accordance with the primary quantity (PRIQTY) value specified.
- The quantity values will always be in a track or a cylinder boundary.
- The quantity values will never be higher than the maximum space expected to be available on a DASD volume, according to user specifications.

To use this tool, you need to have an ISPF edit session on the file containing the DB2 DDL you want to process. Although the CDB2DDL edit macro does not issue a SAVE edit command (and you could always cancel saving the updates introduced by the edit macro), it is strongly suggested that you always back-up the file being edited into another file (or member) before you invoke the CDB2DDL edit macro. Having a saved copy of the DDL will allow you to compare the before and after image of the DDL. It is also recommended that you change your dataset ISPF profile to avoid auto-saving changes when

pressing the END PFKEY. You can do so by issuing the ISPF command 'PROFILE AUTOSAVE OFF'. For additional information, please refer to the *ISPF Edit and Edit macros* manual pertinent to the MVS or OS/390 operating system level at your installation.

It is important to understand how the CDB2DDL edit macro behaves at execution time. Any edit macro behaves like a sub-routine of the ISPF edit session where the macro is invoked. When an edit macro is called the first time, it processes until it returns control to the ISPF edit session. When called a second time, it will not know it had been called before unless some kind of flag or variable had been set the first time it ran. In the case of ISPF Edit, several ISPF variable pools are available to store temporary and/or permanent information that can be accessed by ISPF dialogs and applications, such as an edit macro.

The CDB2DDL edit macro uses the profile variable pool to store variables it uses every time it executes. In particular, it stores the current date, time, and file name being edited, and it uses these variables to control its behaviour at execution time. By using these variables, the edit macro is able to determine whether it is the first time a file is being processed or the process is a re-start. If processing a file for the first time, the edit macro will start checking DDL statements from the top of the file being edited. If re-starting the CDB2DDL edit macro on the current file, the edit macro will resume the editing process on the last DDL statement processed on the previous run of the edit macro. Additional information on this topic can be found in the comments in the CDB2DDL source code.

To invoke the CDB2DDL edit macro, you must be editing (or viewing) the file containing the DB2 DDL. There are five ways to invoke the CDB2DDL edit macro. They are as follows:

- Enter CDB2DDL with no parameters in the command field on line 2 of the edit session display. The edit macro will start executing and it will set the status of the INUSE variable depending on the name of the file being edited, date and time, and elapsed time since last invoked. If the status of INUSE is set to OFF, the macro will display the data entry panels PDB2DDL1 and PDB2DDL2, and will set the INUSE variable to ON. If the status of INUSE is ON, then the edit macro will bypass displaying

the data entry panels and continue editing where it had left off before.

- Enter CDB2DDL RESET in the command field on line 2 of the Edit (or View) session display. Using this option will reset the INUSE variable to OFF and execute the CDB2DDL edit macro as if it had not executed before.
- Enter CDB2DDL SHOW in the command field on line 2 of the Edit (or View) session display. This option is similar to option 1, but it will allow the edit macro to call debugging subroutines not normally invoked. These subroutines will display information about the tokens for the current DDL statement being edited, as well as the contents of the important variables.
- Enter CDB2DDL DEBUG in the command field on line 2 of the Edit (or View) session display. This option is similar to option 1, but it will turn on the REXX traces for this execution of the edit macro. This option is not recommended unless you have a problem or make modifications to the REXX code. Expect many REXX messages.
- Enter CDB2DDL HELP in the command field on line 2 of the Edit (or View) session display. This option will display panel PDB2DDLC, which contains information on how to invoke the edit macro.

Note: be aware that existing changes (manual or by the CDB2DDL edit macro) to the file being edited will remain in place, regardless of which option is chosen. The only way to be sure that no changes are introduced to the file is by issuing the CANCEL command.

After invoking the CDB2DDL edit macro, panel PDB2DDL1 will display. This panel will request you to input the parameters that will be used by the edit macro to review and edit the DDL. After specifying the parameters and pressing Enter, panel PDB2DDL2 will display some information about the DASD device type that was selected. The user will have the opportunity to check these DASD parameters and decide whether it is OK to proceed with them. If not, execution of the CDB2DDL edit macro will terminate.

Description of the parameters specified in the PDB2DDL1 panel are as follows:

1 **Which version of DB2/MVS is this DDL to be used on?** Two versions, 4.1 and 5.1, are supported. The CDB2DDL edit macro will use the DB2 version specified to ensure the DDL does not have incorrect parameters.

2 **Which database default STOGROUP is to be used?** The CDB2DDL edit macro will check whether any CREATE DATABASE or ALTER DATABASE statements found in the DDL have a STOGROUP clause specified. If so, it will check to see whether the parameter for that clause matches the user-specified value. See input field 11 for additional information.

3 **Which database default BPNAME is to be used?** The CDB2DDL edit macro will check whether any CREATE DATABASE or ALTER DATABASE statements found in the DDL has a BPNAME clause specified. If so, it will check to see whether the parameter for that clause matches the user-specified value. See input field 11 for additional information.

4 Which tablespaces are to be allocated using VCAT or STOGROUP? Tablespaces are supported by VSAM linear datasets. These VSAM datasets can be user-specified or DB2-managed. The user-specified VSAM linear datasets are allocated by the user and DB2 needs to know only the high-level qualifier of those datasets, also known as the VCAT name. DB2-managed datasets will be allocated by DB2 itself, using a Storage Group definition, which maps to a VCAT and DASD volumes. Be aware that if you specify VCAT as a parameter for the USING clause, you will not be able to specify the PRIQTY or SECQTY clauses. See input field 11 for additional information.

5 **Which tablespaces default VCAT or STOGROUP name is to be used?** This parameter works in conjunction with input field number 4. It specifies the ‘name’ that will be used for tablespaces allocated via VCAT or the STOGROUP definitions. Refer to the DB2 SQL reference manual for more details about these two options. See input field 11 for additional information.

6 **Which tablespaces default Bufferpool (BP0-BP49 and BP32K-BP32K9)?** The CDB2DDL edit macro will check whether any CREATE TABLESPACE or ALTER TABLESPACE statements have a bufferpool clause specified. If so, it will check to see whether the parameter for the clause matches the user specified value. If no bufferpool clause is found, a clause with the default bufferpool will be inserted in the DDL. See input field 11 for additional information.

7 **Which index spaces are to be allocated using VCAT or STOGROUP?** Similar to input field number 4, except it is for index spaces. See input field 11 for additional information.

8 **Which index spaces default VCAT or STOGROUP name is to be used?** Similar to input field number 5, except it is for index spaces. See input field 11 for additional information.

9 **Which Index spaces default Bufferpool (BP0-BP49)?** Similar to input field number 6, except it is for index spaces. See input field 11 for additional information.

10 **Enter default INDEX type to be used for indexes (1/2).** Starting with DB2/MVS Version 4, indexes can be Type 1 or Type 2. This input field is used to specify the user preference to be checked and supported by the edit macro. See field 11 for additional information.

11 **Do you want to enforce the defaults listed above (Y/N)**? This input field is very important since it tells the edit macro whether (or not) you want to replace the parameters in the DDL with the user-specified options listed above. If set to 'Y', every time the edit macro finds a parameter not matching the default specified by the user, it will replace that parameter with the user-specified values and an informational message will be inserted into the file being edited. If set to 'N", every time the edit macro finds a parameter not matching the default specified by the user, it will introduce an informational message indicating the difference from the default. Note that space and segment size computations will always be performed, regardless of the value for this input field.

12 **Which default type of DASD device is to be used for DB2 Objects?** This input field specifies what type of DASD devices will be used to allocate the DB2 VSAM linear datasets. The DASD type will determine the device geometry to be used for PRIQTY, SECQTY and SEGSIZE computations.

13 **What % of SPACE is expected to be available on DASD volumes (10-100)?** This input field specifies what percentage of the nominal space of a DASD device type will be available at the time the datasets are to be allocated. This percentage is multiplied by the total device capacity, and the result is used to limit the maximum space allocation for the PRIQTY and SECQTY values. If you do not want to be limited by this field, specify 100%.

14 **Do you want the edit macro to STOP on every error (Y/N)?** This input field specifies whether you want the edit macro to stop every time it finds one of the parameters not matching a user-specified value. The edit macro will always stop if it finds a syntax error, or an incorrect parameter, regardless of the value of this input field.

To continue, press Enter after all the above listed input fields are specified. If no errors are found in the input values, the next panel will be displayed.

Panel PDB2DDL2 will display information regarding the DASD devices that will be used, as well as information about the maximum space expected to be available on these devices. Review these values carefully, since they will limit the maximum quantity values to be used for the PRIQTY and SECQTY clauses. A question will be displayed at the bottom of panel PDB2DDL2, asking: 'Do you want to continue with the execution of this edit macro using the above listed values (Y/N)?'

Enter 'N' to end the execution of the CDB2DDL edit macro. Please be aware that existing changes to the file will remain in place. Enter 'Y' to continue with the execution of the edit macro. IF 'Y' was specified, panel PDB2DDL3 will display messages indicating the status of the editing process.

Execution will terminate when all DDL statements have been processed

by the edit macro, whenever a severe error is encountered, or whenever the user decides to interrupt the execution of the edit macro by pressing the Attention key.

The edited file will contain informational comments that will indicate where the CDB2DDL edit macro made changes, or where there are differences between the DDL parameters and the user-specified options. You can issue the 'LOCATE NEXT SPECIAL' ISPF command to reposition the cursor from one informational message to the next. It is strongly suggested that you change one of your function keys' settings to 'LOCATE NEXT SPECIAL' so you can easily navigate the edited DDL.

Even though the REXX edit macro performs quickly, please take note that REXX code can be compiled for greater performance. This may be an appealing option if you have very large DDL files to edit on a regular basis. Compiling REXX code also has the advantage of not allowing changes to the code to be implemented on-the-fly, thus providing a more stable environment for the users. A possible downside of compiling REXX code is that you remove the option of implementing changes on-the-fly. Also note that ISPF panels can be pre-processed for similar reasons, to enhance performance and to disallow updates to them.

CONCLUSION

My main intention in writing the CDB2DDL edit macro was to create a tool that would simplify my work as a DBA, as well as the work of my DBA colleagues. I have spent a significant amount of time editing DDL, retrofitting it to shop standards. The CDB2DDL edit macro will significantly reduce any future time I spend doing DDL editing. I hope it does the same for you.

Developer notes and disclaimer

The CDB2DDL REXX edit macro generates numbered DDL error messages. Some of the errors are the same, but have a different number. This peculiarity is by design in order to allow easier maintenance of the REXX code.

Two sub-routines are at the heart of the CDB2DDL REXX edit macro, the SPLIT_DDL_TOKENS and the CHECK_FOR_BAD_TOKENS routines. These two routines will pre-process and edit the DDL before any clause or parameter is parsed and checked.

A known potential limitation of this edit macro is how the values parameters for create index statements are parsed by the CHECK_CLAUSES and PARSE_INDEX_PARTS subroutines. These subroutines expect the parameters for the values clause to be delimited by parentheses, so if an unbalanced number of them are found, an error message will be generated and execution will stop.

The CDB2DDL edit macro has been extensively tested against DDL generated by Platinum Strategies, Bachman, and ERWIN database modelling tools, and manually generated DDL. Since it has not been tested with DDL generated by other tools, there is always the chance that the edit macro will not behave as intended. Therefore it is important that you always take the precaution of copying your DDL into a save area. It is also recommended that you compare the saved copy with the CDB2DDL macro edited version to ensure that the changes introduced by it were the desired ones. Enjoy it!

```
/* REXX */
'ISREDIT MACRO (PARM1)'  /* SIGNAL TO ISPF THAT THIS IS AN EDIT MAC. */
/********************************************************************/
/* NAME OF EDIT MACRO = CDB2DDL (CHECK DB2 DATA DEFINITION LANGUAGE) */
/* REXX EDIT MACRO BY ANTONIO SALCEDO - 1997-1998-1999-2000         */
/********************************************************************/
/* TO INVOKE THIS EDIT MACRO, YOU MUST BE ON AN "EDIT" OR "VIEW"    */
/* ISPF SESSION.  THIS MACRO ACCEPTS PARAMETERS.  VALID CALLS FROM  */
/* THE COMMAND LINE OF THE "EDIT" OR "VIEW" ISPF SESSION ARE :      */
/* COMMAND ==> "CDB2DDL"                                            */
/* COMMAND ==> "CDB2DDL RES" OR "CDB2DDL RESET"                     */
/* COMMAND ==> "CDB2DDL SHOW"                                       */
/* COMMAND ==> "CDB2DDL DEBUG"                                      */
/* COMMAND ==> "CDB2DDL HELP"                                       */
/* ( NOTE : REMOVE THE DOUBLE QUOTES WHEN INVOKING THE EDIT MACRO )  */
/********************************************************************/
/* PURPOSE OF EDIT MACRO:                                           */
/* THIS EDIT MACRO PROCESSES A FILE CONTAINING THE DDL FOR A DB2/MVS */
/* DATABASE. IT'S MAIN PURPOSE IS TO ENSURE THAT CLAUSES AND THEIR  */
/* PARAMETERS ARE APPROPIATE AND IN ACCORDANCE TO USER SPECIFIED    */
/* DEFAULTS.  IN ADDITION, THE PRIQTY, SECQTY AND SEGSIZE VALUES ARE */
/* CHECKED AND RECOMPUTED TO ENSURE THAT THEY ARE IN ACCORDANCE WITH */
/* THE DASD DEVICE GEOMETRY SELECTED BY THE USER.                   */
```

```
/* SOME OF THE CONCEPTS USED IN THIS REXX EDIT MACRO ARE DESCRIBED   */
/* IN THE ARTICLE 'PECULIARITIES OF THE DB2 CREATE STATEMENT', WHICH */
/* WAS PUBLISHED IN THE DB2 UPDATE ISSUE #72, DATED OCTOBER 1998.    */
/*********************************************************************/
/* INSTALLATION INSTRUCTIONS:                                        */
/* TO USE THIS EDIT MACRO, YOU NEED TO COPY IT INTO A PARTITIONED    */
/* DATASET (PDS) ALLOCATED TO EITHER THE ISPELIB, ISPCLIB OR THE     */
/* ISPEXEC DDNAMES OF YOUR TSO LOGON JCL PROCEDURE.  YOU WILL ALSO   */
/* NEED TO COPY 5 PANELS, PDB2DDL1, PDB2DDL2, PDB2DDL3, PDB2DDLH AND */
/* PDB2DDLI INTO ONE OF THE PARTITIONED DATASETS (PDS) ALLOCATED TO  */
/* THE ISPPLIB DDNAME OF YOUR TSO LOGON JCL PROCEDURE.  AND FINALLY, */
/* YOU NEED TO COPY MEMBER MDDLØØ CONTAINING THE ISPF MESSAGES FOR   */
/* THE PANELS INTO A PDS ALLOCATED TO THE ISPMLIB DDNAME OF YOUR TSO */
/* LOGON JCL PROCEDURE.  AN ALTERNATIVE WAY IS TO USE LIBDEF STMTS.  */
/*********************************************************************/

/*********************************************************************/
/* ALTERNATIVE WAY TO POINT TO THE PANELS AND MESSAGES NEEDED.       */
/* UNCOMMENT THE FOLLOWING STATEMENTS.  CHANGE THE DATASET NAMES.    */
/*********************************************************************/

/* ADDRESS ISPEXEC "LIBDEF ISPPLIB DATASET ID('USERID.ISPPLIB')" */
/* ADDRESS ISPEXEC "LIBDEF ISPMLIB DATASET ID('USERID.ISPMLIB')" */

/*********************************************************************/
/* DB2 (MVS AND OS/39Ø) SPECIFICATIONS FOR V4.1 AND V5.1 THAT ARE    */
/* USED BY THIS MACRO FOR CALCULATING PRIQTY, SECQTY AND SEGSIZE.    */
/*********************************************************************/
/* 1) FOR SIMPLE OR SEGMENTED TABLESPACES, THE MAXIMUM SIZE IS 64    */
/*    GIGA-BYTES.  THE DB2 TABLESPACE IS SUPPORTED BY A SET OF VSAM  */
/*    DATASETS.  THE MAXIMUM SIZE OF THESE VSAM DATASETS CAN BE UP   */
/*    TO TWO (2) GIGA-BYTES.  THUS, 32 VSAM DATASETS WILL BE NEEDED  */
/*    TO SUPPORT A 64 GIGA-BYTE TABLESPACE.                          */
/*-------------------------------------------------------------------*/
/* 2) FOR A PARTITIONED TABLESPLACE, THE MAXIMUM SIZE DEPENDS ON THE */
/*    DB2 RELEASE. V4.1 MAXIMUM SIZE IS 64 GIGA-BYTES.  V5.1 LARGE   */
/*    TABLESPACES CAN BE UP TO 1 TERABYTE OF DATA, SUPPORTED BY 254  */
/*    VSAM DATASETS, EACH ONE OF THEM WITH FOUR GIGA-BYTES IN SIZE.  */
/*-------------------------------------------------------------------*/
/* 3) THE MAXIMUM NUMBER OF EXTENTS OF A DB2 LINEAR VSAM DATASET CAN */
/*    VARY, DEPENDING ON THE RELEASE OF THE OPERATING SYSTEM AND     */
/*    THE DFDSS LEVEL.  PRIOR TO OS/39Ø V2.5, THE MAXIMUM NUMBER OF  */
/*    EXTENTS FOR A VSAM DATASET WAS 119 EXTENTS.  WITH OS/39Ø 2.5   */
/*    AND LATER VERSIONS, THE NUMBER HAS INCREASED TO 251 EXTENTS.   */
/*-------------------------------------------------------------------*/
/* 4) THE MAXIMUM PRIMARY QUANTITY THAT CAN BE USED BY A DB2 OBJECT  */
/*    DEPENDS ON THE OBJECT TYPE :                                   */
/*    - FOR A SIMPLE OR SEGMENTED TABLESPACE, THE MAXIMUM PRIMARY    */
/*      QUANTITY THAT CAN BE USED IS TWO(2) GIGA-BYTES, REGARDLESS   */
/*      OF THE DB2 VERSION.  HOWEVER, BE AWARE THAT YOU STILL CAN    */
```

```
/*      SPECIFY A PRIQTY OF UP TO 4 GIGA-BYTES, THUS WASTING SPACE.  */
/*    - FOR A PARTITIONED TABLESPACE, THE MAXIMUM PRIMARY QUANTITY   */
/*      THAT CAN BE SPECIFIED IS FOUR(4) GIGA-BYTES.  HOWEVER, NOTE  */
/*      THAT THE MAXIMUM SIZE THAT WILL BE USED DEPENDS ON THE DB2   */
/*      VERSION AND THE NUMBER OF PARTITIONS.  DETAILS AS FOLLOWS:   */
/*      - FOR DB2 V4.1, THE MAXIMUM SIZE THAT WILL BE *USED* DEPENDS */
/*        ON THE NUMBER OF PARTITIONS.  IF THE NUMBER OF PARTITIONS  */
/*        IS LESS THAN 16, UP TO 4 GIGA-BYTES WILL BE USED FOR EACH  */
/*        PARTITION.  IF THE NUMBER OF PARTITIONS IS LESS THAN 32,   */
/*        UP TO 2 GIGA-BYTES WILL BE USED FOR EACH PARTITION.  IF    */
/*        MORE THAN 32 PARTITIONS, UP TO 1 GIGA-BYTE FOR PARTITION   */
/*        WILL BE USED.  THIS MAXIMUM *USAGE* IS REGARDLESS OF THE   */
/*        PRIQTY AND SECQTY VALUES SPECIFIED IN THE DDL.             */
/*      - FOR DB2 V5.1, THE MAXIMUM SIZE THAT WILL BE *USED* DEPENDS */
/*        IF THE TABLESPACE IS "LARGE" OR NOT.  IF NOT, THE V4.1     */
/*        RULES APPLY.  IF A "LARGE" TABLESPACE, EACH PARTITION CAN  */
/*        *USE* UP TO 4 GIGA-BYTES REGARDLESS THE # OF PARTITIONS.   */
/*-------------------------------------------------------------------*/
/* 5) THE MAXIMUM SECONDARY QUANTITY THAT CAN BE SPECIFIED FOR A DB2 */
/*    TABLESPACE OR INDEX SPACE IS 128 MEGA-BYTES.                   */
/*********************************************************************/

/*********************************************************************/
/* FOR DEVELOPMENT AND MAINTENANCE PURPOSES, THIS REXX HAS :         */
/* MAIN SECTION : HIGHER LEVEL LOGIC AND CODE FOR THIS EDIT MACRO.   */
/* SECTIONS     : CODE THAT DEALS WITH SPECIFIC DDL STATEMENTS.      */
/* SUB-SECTIONS : THEY GROUP SEVERAL PARAGRAPHS FOR THE SAME PURPOSE */
/* PARAGRAPHS   : SMALL PIECES OF CODE THAT HAVE SPECIFIC PURPOSE.   */
/* SUB-ROUTINES : COMMON CODE THAT CAN BE REUSED MULTIPLE TIMES, OR  */
/*                CODE THAT HAS BEEN SEGREGATED FOR READABILITY.     */
/*********************************************************************/

/*********************************************************************/
/*  DOCUMENTATION SECTION FOR IMPORTANT VARIABLES USED IN THIS REXX  */
/*-------------------------------------------------------------------*/
/* THERE ARE SIX VARIABLES USED TO PARSE AND EDIT THE USING CLAUSE.  */
/* THESE VARIABLES ARE CHECKED AGAINST THE DEFAULTS PROVIDED BY THE  */
/* USER IN THE INPUT PANEL.                                          */
/*-------------------------------------------------------------------*/
/* DFTSSTOR AND DFIXSTOR : THESE TWO VARIABLES STORE THE VALUES THAT */
/* DESCRIBE IF THE VSAM LINEAR DATASETS FOR TABLESPACES AND INDEX    */
/* SPACES ARE TO BE MANAGED BY DB2 OR BY THE USER.  VALUES THAT CAN  */
/* BE SPECIFIED ARE:  VCAT(USER) AND STOGROUP(DB2 MANAGED).          */
/*-------------------------------------------------------------------*/
/* DFTSNAME : VARIABLE CONTAINS THE DEFAULT VCAT/STOGROUP TO BE USED */
/* FOR TABLESPACES.  VALUE IS SET IN PANEL PDB2DDL1                  */
/*-------------------------------------------------------------------*/
/* DFIXNAME : VARIABLE CONTAINS THE DEFAULT VCAT/STOGROUP TO BE USED */
/* FOR INDEXSPACES.  VALUE IS SET IN PANEL PDB2DDL1                  */
/*-------------------------------------------------------------------*/
```

```
/* DFDBSTOG :   CONTAINS THE NAME OF THE STOGROUP USED ON THE CREATE */
/* DATABASE STATEMENT. IF NONE SPECIFIED AND THE DFTSSTOR VARIABLE   */
/* EQUALS 'STOGROUP', THEN THIS EDIT MACRO WILL INSERT A STOGROUP    */
/* STATEMENT USING THE CONTENTS OF THE DFTSNAME VARIABLE.            */
/*-------------------------------------------------------------------*/
/* U_STOR : VARIABLE CONTAINS THE TYPE OF ALLOCATION METHOD USED     */
/* ON THE CURRENT DDL STATEMENTS BEING PARSED.                       */
/*-------------------------------------------------------------------*/
/* U_NAME : VARIABLE CONTAINS THE NAME OF THE VCAT OR THE STORAGE    */
/* GROUP SPECIFIED ON THE CURRENT DDL STATEMENTS BEING PARSED.       */
/*********************************************************************/

/*-------------------------------------------------------------------*/
/*  SECTION :   INITIALIZATION AND TRACE PROCESSING                  */
/*-------------------------------------------------------------------*/
PARM1 = TRANSLATE(PARM1)                       /* TRANSLATE PARAMETER */
SELECT
   WHEN PARM1 = ''      THEN NOP               /* NO PROCESS          */
   WHEN PARM1 = 'RES'   THEN NOP               /* NO PROCESS          */
   WHEN PARM1 = 'RESET' THEN NOP               /* NO PROCESS          */
   WHEN PARM1 = 'SHOW'  THEN NOP               /* NO PROCESS          */
   WHEN PARM1 = 'DEBUG' THEN TRACE I           /* TURN TRACE ON       */
   OTHERWISE DO                                /* INCLUDES 'HELP'     */
      'ISPEXEC DISPLAY PANEL(PDB2DDLC)'        /* SHOW COMMAND HELP   */
      EXIT                                     /* EXIT EDIT MACRO     */
      END                                      /* END OF OTHERWISE    */
   END                                         /* END OF SELECT STMT  */

/*-------------------------------------------------------------------*/
/*  SET CONSTANTS TO BE USED BY THIS REXX EDIT MACRO.                */
/*-------------------------------------------------------------------*/
YVAL = 'Y'                                     /* SET CONSTANT        */
BLANK = " "                                    /* SET BLANK SPACE     */
SC_HEX_VAL = C2X(';')   /* SET HEX VALUE OF SEMI-COLON TO SEARCH FOR */
NUMPARTS = Ø            /* SET SPECIAL LOGIC CONTROL VARIABLE TO Ø.  */
TRKCAP = Ø; CYLCAP = Ø; DEVCAP = Ø; USERCAP = Ø;   /* INIT DASD VARS */
MIN_DASD_ALLOC = 'PRIQTY XX SECQTY YY'         /* SET TO DUMMY VALUES */

/*-------------------------------------------------------------------*/
/*  TRY TO GET PREVIOUSLY SAVED VALUES FROM THE PROFILE POOL         */
/*-------------------------------------------------------------------*/
'ISPEXEC VGET ( SETDFLAG DB2VER INITLCNT ) PROFILE'
'ISPEXEC VGET ( DFDBBPNM DFTSBPNM DFIXBPNM DFDBSTOG ) PROFILE'
'ISPEXEC VGET ( DFTSSTOR DFTSNAME DFIXSTOR DFIXNAME ) PROFILE'
'ISPEXEC VGET ( DFIXTYPE DFDASD   DFPERC   STOPFLAG ) PROFILE'
'ISPEXEC VGET ( INUSE    LASTDATE LASTHOUR LASTMINU ) PROFILE'
'ISPEXEC VGET ( CMDSNAME CMMEMBER EDITLINE) PROFILE'

/*-------------------------------------------------------------------*/
/*  PARAGRAPH : FIND IF WE ARE CURRENTLY USING THE EDIT REXX EXEC OR */
```

```
/*  NOT, AS INDICATED BY THE INUSE VARIABLE.  IF THE INUSE VARIABLE  */
/*  IS ON, CHECK THAT A) WE ARE EDITING THE SAME FILE AND THAT B) .  */
/*  THE LAST TIME THE EDIT MACRO WAS LESS THAN 1Ø MINUTES AGO.       */
/*  ALSO, IF PARM = 'RES' OR FILE IS DIFFERENT, THEN SET INUSE=NO.   */
/*-------------------------------------------------------------------*/

'ISPEXEC VGET (ZDATE ZTIME)'
PARSE VAR ZTIME CURRHOUR ":" CURRMINU .
CURRDATE = ZDATE
'ISREDIT (TDSN) = DATASET'
'ISREDIT (TMEM) = MEMBER'
IF ( INUSE = 'YES' ) THEN DO                  /* IF INUSE = 'YES'    */
   IF ( PARM1 = 'RES' ) | ( PARM1 = 'RESET' ) THEN INUSE = 'NO'
   IF ( (CMDSNAME \= TDSN) | (CMMEMBER \= TMEM) ) THEN INUSE = 'NO'
   IF ( ( CURRDATE = LASTDATE ) & ( INUSE = 'YES' ) ) THEN DO
      CURRNUM = ( CURRHOUR * 6Ø ) + CURRMINU
      LASTNUM = ( LASTHOUR * 6Ø ) + LASTMINU
      IF ( CURRNUM < ( LASTNUM + 1Ø ) ) THEN NOP     /* CUSTOMIZABLE */
      ELSE INUSE = 'NO'
      END
   ELSE INUSE = 'NO'                          /* DIFFERENT DATE      */
   IF ( INUSE = 'YES' ) THEN CALL RESET_LAST_SPECIAL_MESSAGE
   END                                        /* END IF INUSE = YES  */

   /*----------------------------------------------------------------*/
   /*  DEPENDING ON VALUE OF INUSE VARIABLE, EITHER CALL THE ROUTINE */
   /*  THAT COMPUTES THE DASD PARAMETERS OR PROCEED TO DISPLAY THE   */
   /*  PANEL PDB2DDL1 TO OBTAIN ALL OF THE USER INPUT PARAMETERS.    */
   /*----------------------------------------------------------------*/
IF ( INUSE = 'YES' ) THEN CALL COMPUTE_DASD_PARMS
ELSE DO                                    /* NOT IN USE, FIRST TIME  */
   /* RESET MESSAGES & NOTES IN THE FILE CURRENTLY BEING EDITED      */
   'ISREDIT RESET SPECIAL'                 /* RESET SPECIAL LINES     */
   INUSE = 'YES'
   'ISREDIT (INITLCNT) = LINENUM ' .ZLAST /* GET THE INITIAL # LINES */
   'ISPEXEC DISPLAY PANEL(PDB2DDL1)'       /* HELP PANEL IS PDB2DDLH  */
   EXITCNTL = RC
   'ISPEXEC VPUT ( SETDFLAG DB2VER INITLCNT )            PROFILE'
   'ISPEXEC VPUT ( DFDBBPNM DFTSBPNM DFIXBPNM DFDBSTOG ) PROFILE'
   'ISPEXEC VPUT ( DFTSSTOR DFTSNAME DFIXSTOR DFIXNAME ) PROFILE'
   'ISPEXEC VPUT ( DFIXTYPE DFDASD   DFPERC   STOPFLAG ) PROFILE'
   IF (EXITCNTL=8) THEN DO          /* PF3 KEY WAS PRESSED AT PANEL. */
      ZEDSMSG= 'ENDING EDIT MACRO'
      ZEDLMSG= 'YOU HAVE PRESSED PF3 AND EXIT THE CDB2DDL EDIT MACRO'
      'ISPEXEC SETMSG MSG(ISRZØØ1)'
      INUSE = 'NO'
      END                           /* END OF EXITCNTL=8 STATEMENT   */
   /* SAVE CURRENT INUSE, LASTDATE, LASTHOUR, LASTMINU VARIABLES     */
   LASTDATE = CURRDATE              /* SET VALUE TO CURRENT DATE     */
   LASTHOUR = CURRHOUR              /* SET VALUE TO CURRENT HOUR     */
```

```
   LASTMINU = CURRMINU              /* SET VALUE TO CURRENT MINUTES  */
   CMDSNAME = TDSN                  /* SET VALUE TO CURRENT DATASET  */
   CMMEMBER = TMEM                  /* SET VALUE TO CURRENT MEMBER   */
   EDITLINE = 1                     /* SET VALUE TO FIRST LINE NO    */
   'ISPEXEC VPUT ( INUSE LASTDATE LASTHOUR LASTMINU ) PROFILE'
   'ISPEXEC VPUT ( CMDSNAME CMMEMBER EDITLINE ) PROFILE'
   IF (EXITCNTL=8) THEN EXIT        /* EXIT THIS EDIT REXX MACRO     */

   /*-------------------------------------------------------------------*/
   /*  PARAGRAPH :  COMPUTE THE DASD PARAMETERS TO BE USED BY MACRO  */
   /*-------------------------------------------------------------------*/
   CALL COMPUTE_DASD_PARMS

   /*-------------------------------------------------------------------*/
   /* DISPLAY PANEL PDB2DDL2 WITH INFORMATION ON DASD CAPACITY.  SET */
   /* VARIABLE PARMOKAY IN PANEL PDB2DDL2 TO CONTROL EXIT FROM MACRO */
   /*-------------------------------------------------------------------*/
   PARMOKAY = YVAL                  /* PREINITIALIZE PARMOKAY VAR.   */
   'ISPEXEC DISPLAY PANEL(PDB2DDL2)'      /* HELP PANEL IS PDB2DDLI  */
   IF PARMOKAY \= YVAL THEN DO      /* CHOSE NOT TO CONTINUE. EXIT.  */
      ZEDSMSG= 'EXITING EDIT MACRO'
      ZEDLMSG= '  EXITING THE EDIT MACRO AFTER REVIEWING DASD PARMS'
      'ISPEXEC SETMSG MSG(ISRZØØ1)'
      INUSE = 'NO'
      EXIT                          /* EXIT THIS EDIT REXX MACRO     */
      END                           /* END IF PARMOKAY \= YVAL       */
   END                              /* END OF ELSE STATEMENT         */

/*********************************************************************/
/*  MAIN-SECTION : LOOP THRU DDL LOOKING FOR VALID DDL STATEMENT.    */
/*  EDITLINE VALUE COULD CONTAIN EITHER 1 OR THE VALUE WHERE THE     */
/*  EDIT MACRO LEFT THE EDITING THE LAST TIME.                       */
/*********************************************************************/
CL1 = EDITLINE                  /* SET CL1 TO CONTENTS OF EDITLINE */
CL2 = Ø                         /* INITIALIZE CL2 TO ZERO          */
CALL FIND_NEXT_DDL_STMT         /* SET CONDITION CODE AFTER FIND   */

DO WHILE ( LOOPCNTL = Ø )       /* MAIN EDIT MACRO LOOP BEGINS     */

   /*-------------------------------------------------------------------*/
   /*  PARAGRAPH :  ENSURE 'CREATE' IS FIRST TOKEN IN LINE.          */
   /*  SAVE CURRENT CURSOR POSITION ON VARIABLES CL1,CC1. CHECK THAT */
   /*  THE FIRST TOKEN IS A CREATE STMT. IF NOT, THEN ISSUE A FIND   */
   /*  FOR THE NEXT 'CREATE' WORD AND GO TO THE TOP OF THE LOOP.     */
   /*-------------------------------------------------------------------*/
   'ISREDIT (CL1,CC1) = CURSOR'          /* GET CURRENT LINE NUMBER  */
   'ISREDIT (LNX) = LINE' CL1            /* MAKE LINE CONTENTS AVAIL */

   /*-------------------------------------------------------------------*/
   /*  PARAGRAPH :  FIND THE ';' TOKEN DELIMITING THE END OF THE     */
```

```
/*             DDL STRING.  IF NOT FOUND, LEAVE EXIT THE LOOP.   */
/*---------------------------------------------------------------*/
'ISREDIT FIND X"'SC_HEX_VAL'" NEXT'        /* FIND NEXT ; TOKEN   */
IF ( RC = Ø ) THEN 'ISREDIT (CL2,CC2) = CURSOR'
ELSE DO                                    /* NO MORE ; WHERE FOUND    */
   LOOPCNTL = 1                            /* SET FLAG TO EXIT LOOP    */
   ITERATE                                 /* EXIT CURRENT LOOP RUN    */
   END                                     /* END OF ELSE STATEMENT    */

/*---------------------------------------------------------------*/
/*  PARAGRAPH : STORE DDL INTO DDLSTR VARIABLE. CHECK FOR SPECIAL */
/*  CHARACTERS IN THE DDL STR AND ISSUE ERROR MESSAGES IF FOUND.  */
/*---------------------------------------------------------------*/
DDLSTR = ''                               /* RESET DDLSTR TO SPACES    */
TL1 = CL1                                 /* SET TEMP LINE COUNTER     */
DO WHILE ( TL1 <= CL2 )                   /* LOOP THRU THE DDL LINES   */
   'ISREDIT (LNX) = LINE' TL1             /* MAKE LINE CONTENTS AVAIL  */
   DDLSTR = DDLSTR || ' ' || STRIP(LNX)  /* JOIN DDL AFTER STRIP  */
   TL1 = TL1 + 1                          /* INCREASE LINE COUNTER     */
   END                                    /* END OF STORE SQL SECTION  */

/*---------------------------------------------------------------*/
/*  PARAGRAPH : SAVE CURRENT LINE POSITION FOR RESTART PURPOSES.  */
/*---------------------------------------------------------------*/
EDITLINE = CL1                            /* SET EDITLINE = CL1        */
'ISPEXEC VPUT ( EDITLINE ) PROFILE'  /* STORE IN PROFILE POOL.    */

/*****************************************************************/
/*  SECTION : PROCESS 'CREATE DATABASE' STATEMENT.               */
/*****************************************************************/
/*  FIND IF THE CREATE STATEMENT IS FOR A DATABASE.  IF SO, THEN  */
/*  LOOK FOR THE DIFFERENT CLAUSES OF THE CREATE DB STMT.  ALSO,  */
/*  IF CLAUSES THAT HAVE DEFAULT VALUES ARE MISSING, THE EDIT     */
/*  MACRO WILL INSERT THEM INTO THE DDL.                          */
/*****************************************************************/
IF ( WORDPOS('CREATE',DDLSTR) = 1 ) & ,
   ( WORDPOS('DATABASE',DDLSTR) = 2 ) THEN DO

   /*-------------------------------------------------------------*/
   /*  PARAGRAPH : DISPLAY SCREEN. SPLIT TOKENS & COUNT WORDS.    */
   /*-------------------------------------------------------------*/
   CALL DISPLAY_STATUS 'CREATE DATABASE'
   BADTK = 'NO'                           /* SET BAD TOKEN FLAG = NO  */
   CALL SPLIT_DDL_TOKENS                  /* SEPARATE TOKENS IN DDL   */
   IF ( BADTK = YVAL ) THEN DO            /* CHECK IF BADTK WAS SET Y */
      CALL FIND_NEXT_DDL_STMT             /* FIND NEXT VALID DDL STMT */
      ITERATE                             /* GO TO TOP OF LOOP        */
      END
   DDLWORDS = WORDS(DDLSTR)               /* COUNT NUMBER OF WORDS    */
```

```
/*-------------------------------------------------------------*/
/*  PARAGRAPH : CHECK CLAUSES FOR THE CREATE DATABASE STMT.    */
/*-------------------------------------------------------------*/
DBNAME = WORD(DDLSTR,3)                     /* GET DBNAME PARM */
CALL CHECK_VALID_NAME DBNAME, 'DATABASE'    /* VALIDATE NAME   */
CALL CHECK_CLAUSES 'CREATE','DATABASE'      /* VALIDATE DDL    */

/*-------------------------------------------------------------*/
/*  PARAGRAPH : PARSE/VALIDATE BUFFERPOOL IN CREATE DB STMT.   */
/*  IF CLAUSE IS MISSING, INSERT CLAUSE IN THE DDL.            */
/*-------------------------------------------------------------*/
BPWORD = WORDPOS('BUFFERPOOL',DDLSTR)
IF ( BPWORD > Ø ) THEN DO
   PARSE VAR DDLSTR . 'BUFFERPOOL' DB_BPNAME .
   /* VALIDATE THE BUFFERPOOL NAME.  CHECK IF ; WAS AFTER IT   */
   CALL CHECK_BUFFERPOOL_PARM DB_BPNAME
   BPNAME = STRIP(DB_BPNAME,B,';')
   IF ( BPNAME \= DFDBBPNM ) THEN DO
      'ISREDIT CURSOR = ' CL1 Ø
      'ISREDIT FIND BUFFERPOOL NEXT'
      'ISREDIT FIND "'DB_BPNAME'" NEXT WORD'
      'ISREDIT (XL1,XC1) = CURSOR'
      IF ( SETDFLAG = YVAL ) THEN DO      /* CHANGE TO DEFAULT */
         'ISREDIT (LNX) = LINE' XL1
         PARSE VAR LNX F_TXT (DB_BPNAME) E_TXT
         LNX = F_TXT DFDBBPNM
         CALL INSERT_LINE 'SAME', XL1, LNX
         IF ( E_TXT \= '' ) THEN DO       /* SPLIT THE LINE    */
            LNX = E_TXT
            CALL INSERT_LINE 'AFTER', XL1, LNX
            END
         NT = 'BPNAME NAME' BPNAME 'CHANGED TO DEFAULT' DFDBBPNM
         CALL WRITE_MSG NT, XL1
         END
      ELSE DO
         NT = BPNAME 'BUFFERPOOL NOT = TO USER DEFAULT' DFDBBPNM
         CALL WRITE_NOTE NT, XL1
         END                              /* END SETFLAG ?     */
      END                                 /* END DB_BPNAME ?   */
   END                                    /* END IF BPWORD > Ø */
ELSE DO                                   /* INSERT DEFAULT BP */
   'ISREDIT CURSOR = ' CL1 Ø              /* REPOSITION CURSOR */
   'ISREDIT FIND "'DBNAME'" NEXT WORD'    /* FIND DATABASE POS */
   'ISREDIT FIND '''BLANK''' NEXT'        /* FIND NEXT BLANK   */
   'ISREDIT (XL1,XC1) = CURSOR'           /* SAVE CURSOR POS   */
   'ISREDIT TSPLIT' XL1 XC1               /* SPLIT THE LINE    */
   TLX = 'BUFFERPOOL 'DFDBBPNM            /* SET NEW LINE CONT */
   CALL INSERT_LINE 'AFTER', XL1, TLX     /* INSERT NEW LINE   */
   NT = 'USER DEFAULT BUFFERPOOL' DFDBBPNM ' INSERTED IN DDL'
   CALL WRITE_MSG NT, ( XL1 + 1 )
```

```
   END                                      /* END ELSE BPWORD>Ø */

/*-------------------------------------------------------------*/
/*  PARAGRAPH : PARSE/VALIDATE STOGROUP CLAUSE IN CREATE DB.   */
/*  IF SETDFLAG IS YES, THEN CHANGE THE DDL TO USE DEFAULT.    */
/*-------------------------------------------------------------*/
STOGPOS = WORDPOS('STOGROUP',DDLSTR)
IF ( STOGPOS \= Ø ) THEN DO          /* STOGROUP KEYWORD FOUND */
   STOGNAME = WORD(DDLSTR,(STOGPOS+1))
   CALL CHECK_VALID_NAME STOGNAME, 'STOGROUP' /* VALIDATE NAME */
   STOGNAME = STRIP(STOGNAME,B,';')
   IF ( STOGNAME \= DFDBSTOG ) THEN DO
      'ISREDIT CURSOR = ' CL1 Ø
      'ISREDIT FIND STOGROUP NEXT'
      'ISREDIT FIND "'STOGNAME'" NEXT WORD'
      'ISREDIT (XL1,XC1) = CURSOR'
      IF ( SETDFLAG = YVAL ) THEN DO      /* CHANGE TO DEFAULT */
         'ISREDIT (LNX) = LINE' XL1
         /* CHECK TO SEE IF STOGROUP CLAUSE IS IN SAME LINE    */
         IF (WORDPOS('STOGROUP',LNX) > Ø ) THEN DO
            PARSE VAR LNX F_TXT 'STOGROUP' (STOGNAME) E_TXT
            LNX = F_TXT 'STOGROUP' DFDBSTOG
            END
         ELSE DO                 /* NO, IT IS NOT IN SAME LINE */
            PARSE VAR LNX F_TXT (STOGNAME) E_TXT
            LNX = F_TXT DFDBSTOG
            END
         CALL INSERT_LINE 'SAME', XL1, LNX
         IF ( E_TXT \= '') THEN DO        /* SPLIT THE LINE    */
            LNX = E_TXT
            CALL INSERT_LINE 'AFTER', XL1, LNX
            END
         NT = 'STOGROUP NAME' STOGNAME,
              'CHANGED TO DEFAULT' DFDBSTOG
         CALL WRITE_MSG NT, XL1
         END
      ELSE DO
         NT = 'USING STOGROUP' STOGNAME', NOT DEFAULT' DFDBSTOG
         CALL WRITE_NOTE NT, XL1
         END                              /* IF SETDFLAG ?     */
      END                                 /* IF STOGNAME ?     */
   END                                    /* END IF STOGPOS\=Ø */
ELSE DO           /* NO STOGROUP CLAUSE FOUND.  INSERT DEFAULT */
   'ISREDIT CURSOR = ' CL1 Ø              /* REPOSITION CURSOR */
   'ISREDIT FIND "'DBNAME'" NEXT WORD'    /* FIND DATABASE POS */
   'ISREDIT FIND '''BLANK''' NEXT'        /* FIND NEXT BLANK   */
   'ISREDIT (XL1,XC1) = CURSOR'           /* SAVE CURSOR POS   */
   TLX = 'STOGROUP ' DFDBSTOG             /* USE PANEL VALUE   */
    CALL INSERT_LINE 'AFTER', XL1, TLX    /* INSERT NEW LINE   */
   NT = 'USER DEFAULT STOGROUP' DFDBSTOG 'INSERTED IN DDL'
```

```
      CALL WRITE_MSG NT, ( XL1 + 1 )
      END                              /* END OF ELSE IF STOGPOS/=Ø */
   END                                 /* END OF CREATE DB SECTION  */

/*****************************************************************/
/*  SECTION : PROCESS 'CREATE TABLESPACE' STATEMENTS.            */
/*****************************************************************/
IF ( ( WORDPOS('CREATE',DDLSTR)     = 1 ) & ,
     ( WORDPOS('TABLESPACE',DDLSTR) = 2 ) ) | ,            /* OR */
   ( ( WORDPOS('CREATE',DDLSTR)     = 1 ) & ,
     ( WORDPOS('LARGE',DDLSTR)      = 2 ) & ,
     ( WORDPOS('TABLESPACE',DDLSTR) = 3 ) ) THEN DO

   /*-------------------------------------------------------------*/
   /*  PARAGRAPH : DISPLAY SCREEN. SPLIT TOKENS & COUNT WORDS.    */
   /*-------------------------------------------------------------*/
   CALL DISPLAY_STATUS 'CREATE TABLESPACE'
   BADTK = 'NO'                        /* SET BAD TOKEN FLAG = NO  */
   CALL SPLIT_DDL_TOKENS               /* SEPARATE TOKENS IN DDL   */
   IF ( BADTK = YVAL ) THEN DO         /* CHECK IF BADTK WAS SET Y */
      CALL FIND_NEXT_DDL_STMT          /* FIND NEXT VALID DDL STMT */
      ITERATE
      END
   DDLWORDS = WORDS(DDLSTR)           /* COUNT THE NUMBER OF WORDS */

   /*-------------------------------------------------------------*/
   /*  PARAGRAPH : SEEK FOR SPECIAL CLAUSES IN THE CREATE STMT.   */
   /*-------------------------------------------------------------*/
   BPNMPOS  = WORDPOS('BUFFERPOOL',DDLSTR)     /* FIND BUFFERPOOL */
   USINGPOS = WORDPOS('USING',DDLSTR)          /* FIND USING      */
   NPRTPOS  = WORDPOS('NUMPARTS',DDLSTR)       /* FIND NUMPARTS   */
   SEGMPOS  = WORDPOS('SEGSIZE',DDLSTR)        /* FIND SEGSIZE    */
   LCKPPOS  = WORDPOS('LOCKPART',DDLSTR)       /* FIND LOCKPART   */

   /*-------------------------------------------------------------*/
   /*  PARAGRAPH: GET NUMPARTS AND VALIDATE THAT IT IS NUMERIC.   */
   /*  ALSO, GET THE LINE NUMBER WHERE NUMPARTS CLAUSE IS FOUND.  */
   /*-------------------------------------------------------------*/
   IF ( NPRTPOS > Ø ) THEN DO
      NUMPARTS = WORD(DDLSTR,(NPRTPOS+1))   /* GET NUMPARTS VALUE */
      CALL CHECK_NUMPARTS_PARM NUMPARTS     /* CALL CHECK SUB-RTN */
      PARTFLAG = 'YES'                      /* SET PARTFLAG = YES */
      END
   ELSE DO
      NUMPARTS = Ø                          /* SET NUMPARTS = Ø   */
      PARTFLAG = 'NO'                       /* SET PARTFLAG = NO  */
      END

   /*-------------------------------------------------------------*/
   /*  PARAGRAPH : CHECK THAT BOTH SEGSIZE AND NUMPARTS ARE NOT   */
```

```
/*              SPECIFIED. IF SO, ISSUE ERROR AND EXIT.        */
/*-------------------------------------------------------------*/
IF ( ( NPRTPOS > Ø ) & ( SEGMPOS > Ø ) ) THEN DO
   LM = 'DDL ERROR # 1: NUMPARTS & SEGSIZE - MUTUALLY EXCLUSIVE'
   SM = 'DDL ERROR # 1'
   CALL DDLERROR YVAL, CL1, 'NUMPARTS', 'SEGSIZE', 1
   END

/*-------------------------------------------------------------*/
/*  PARAGRAPH : CHECK CLAUSES FOR THE CREATE TABLESPACE STMT.  */
/*-------------------------------------------------------------*/
CALL CHECK_CLAUSES 'CREATE','TABLESPACE'

/*-------------------------------------------------------------*/
/*  PARAGRAPH : CHECK IF CREATING SIMPLE TABLESPACE.           */
/*-------------------------------------------------------------*/
IF ( ( NPRTPOS = Ø ) & ( SEGMPOS = Ø ) ) THEN DO
   NT = 'YOU ARE USING A SIMPLE TABLESPACE'
   CALL WRITE_NOTE NT, CL1
   NT = 'CONSIDER USING A SEGMENTED OR PARTITIONED TABLESPACE'
   CALL WRITE_NOTE NT, CL1
   END

/*-------------------------------------------------------------*/
/*  SUB-SECTION: CHECK THE FIRST USING CLAUSE FOR PART. TS.    */
/*-------------------------------------------------------------*/
/*  DOCUMENTATION : FOR PARTITIONED TABLESPACES THERE CAN BE A */
/*  USING CLAUSE FOR EACH PARTITION. THE USING CLAUSE CAN BE   */
/*  EXPLICITLY CODED FOR EACH PARTITION, FOR THE WHOLE         */
/*  TABLESPACE, OR BOTH.  IF NO USING CLAUSE IS SPECIFIED AT   */
/*  ANY LEVEL, DB2 WILL USE THE USING CLAUSE (EITHER IMPLICIT  */
/*  OR EXPLICIT) OF THE CREATE DATABASE DEFINITION.            */
/*-------------------------------------------------------------*/
/*  THIS SUB-SECTION WILL ENSURE THAT THERE IS AN EXPLICITLY   */
/*  CODED USING CLAUSE IN THE CREATE TABLESPACE STATEMENT FOR  */
/*  ALL TYPES OF TABLESPACES.  TWO PARAGRAPHS ARE USED, ONE    */
/*  FOR PARTITIONED TABLESPACES & ANOTHER FOR NON-PARTITIONED  */
/*  TABLESPACES.  IF NO USING CLAUSE IS FOUND, THE PARAGRAPHS, */
/*  WILL INSERT AN EXPLICIT USING CLAUSE USING THE DEFAULT     */
/*  VALUES SPECIFIED BY THE USER IN THE PDB2DDL1 PANEL.        */
/*-------------------------------------------------------------*/
IF ( NPRTPOS > Ø ) THEN DO           /* IT IS PARTITIONED TS   */
   IF ( ( USINGPOS > NPRTPOS ) | ( USINGPOS = Ø ) ) THEN DO
      'ISREDIT CURSOR = ' CL1 Ø      /* POSITION CURSOR AT TOP */
      'ISREDIT FIND NUMPARTS NEXT'   /* FIND TOKEN TO SPLIT AT */
      'ISREDIT (XL1,XC1) = CURSOR'   /* GET CURRENT LINE POS   */
      'ISREDIT (LNX) = LINE' XL1     /* MAKE CONTENTS AVAILABLE*/
      PARSE VAR LNX F_TOKEN .        /* GET FIRST TOKEN IN LN  */
      IF \( F_TOKEN = 'NUMPARTS' ) THEN DO
         'ISREDIT TSPLIT' XL1 XC1    /* NOT FIRST, SPLIT LINE  */
```

```
         XL1 = XL1 + 1              /* INCREASE LINE BY ONE   */
         END                        /* END OF NOT EQUAL       */
      /* INSERT THE USING CLAUSE BEFORE THE CURRENT LINE.     */
      TLX = 'USING' DFTSSTOR DFTSNAME
      CALL INSERT_LINE 'BEFORE', XL1, TLX
      NT = 'USING CLAUSE INSERTED WITH USER DEFAULT VALUES'
      CALL WRITE_MSG NT, XL1        /* WRITE INFO NOTE        */
      IF ( DFTSSTOR = 'STOGROUP' ) THEN DO
         TLX = MIN_DASD_ALLOC       /* SET LINE TO INSERT     */
         CALL INSERT_LINE 'AFTER', XL1, TLX
         NT = 'MINIMUM PRIQTY AND SECQTY AMOUNTS INSERTED'
         CALL WRITE_MSG NT, (XL1+1)
         END
      /* DDL HAS BEEN MODIFIED. SINCE NEW PARMS HAVE BEEN      */
      ITERATE          /* INSERTED, RESTART DDL EDITING       */
      END                           /* END                    */
   END  /* END OF PARAGRAPH FOR USING CLAUSE FOR PART. TSPACES */

/*-------------------------------------------------------------*/
/*  PARAGRAPH : PARSE USING CLAUSE FOR NON-PART TABLESPACE.    */
/*  IF USING CLAUSE IS NOT SPECIFIED, THEN ATTEMPT TO FIND     */
/*  A GOOD PLACE TO INSERT IT.  IF CAN'T FIND A PLACE WHERE TO */
/*  INSERT THE USING CLAUSE, THEN DO IT BEFORE THE SEMI-COLON. */
/*-------------------------------------------------------------*/
IF ( NPRTPOS = Ø ) THEN DO          /* NOT PARTITIONED TS.    */
   IF ( USINGPOS \=Ø ) THEN DO
      CNTUSING = Ø                  /* SET VARIABLE TO ZERO   */
      CURPOS = USINGPOS             /* SET CURSOR POSITION    */
      CALL PARSE_USING_PARM         /* SET U_STOR AND U_NAME  */
      END
   IF ( USINGPOS = Ø ) THEN DO       /* WE NEED TO FIND A SPOT */
      TKNPOS = Ø                    /* SET VARIABLE TO ZERO   */
      /* DO SEARCH FOR 'IN DBNAME' TOKENS.                     */
      IF ( TKNPOS = Ø ) THEN DO      /* IF NONE FOUND, SEEK    */
         TKWORD = 'IN'              /* SET TEMP KEYWORD       */
         TKNPOS = WORDPOS(TKWORD,DDLSTR)      /* LOOK FOR IT   */
         IF ( TKNPOS \=Ø ) THEN TOKEN2 = WORD(DDLSTR,(TKNPOS+1))
         END
      /* DO SEARCH FOR 'CREATE' TOKEN.  USE TABLESPACE NAME AS */
      /* THE SECOND TOKEN TO POSITION THE CURSOR FOR EDITING.  */
      IF ( TKNPOS = Ø ) THEN DO      /* IF NONE FOUND, SEEK    */
         TKWORD = 'CREATE'          /* SET TEMP KEYWORD       */
         TKNPOS = WORDPOS(TKWORD,DDLSTR)      /* GET POSITION  */
         IF ( TKNPOS \=Ø ) THEN TOKEN2 = WORD(DDLSTR,(TKNPOS+2))
         END
      /* COULD NOT FIND A TOKEN.          ISSUE ERROR AND STOP  */
      IF ( TKNPOS = Ø ) THEN DO      /* IF NONE, ISSUE ERROR   */
         LM = 'LOGIC ERROR 1: COULD NOT LOCATE PLACE TO INSERT'
         SM = 'LOGIC ERROR 1'
         CALL DDLERROR YVAL, CL1, 'CREATE', '' , 1
```

```
         END
      /* NOW THAT WE FOUND A KEYWORD IN THE DDL, REPOSITION    */
      /* THE CURSOR AFTER THE TKWORD TOKEN2 WORDS AND SPLIT    */
      /* THE LINE SO A NEW LINE CAN BE INSERTED.               */
      'ISREDIT CURSOR = ' CL1 Ø            /* REPOSITION CURSOR*/
      'ISREDIT FIND "'TKWORD'" NEXT'       /* FIND TEMP KEYWORD*/
      'ISREDIT FIND "'TOKEN2'" NEXT'       /* FIND TOKEN2      */
      'ISREDIT FIND '''BLANK''' NEXT'      /* POS AT NEXT SPACE*/
      'ISREDIT (XL1,XC1) = CURSOR'         /* GET POS IN LINE  */
      'ISREDIT (LNX) = LINE' XL1           /* GET LINE CONTENT */
      PARSE VAR LNX F_TXT (TKWORD) (TOKEN2) E_TXT
      IF ( E_TXT \= '' ) THEN DO           /* IF OTHER STUFF   */
         'ISREDIT TSPLIT' XL1 (XC1-1)      /* = TKWORD, THEN   */
         XL1 = XL1 + 1                     /* SPLIT THE LINE   */
         END                               /* END F_TOKEN CHKC */
      /* PROCEED TO INSERT THE USING CLAUSE AFTER CURRENT LINE */
      TLX = 'USING 'DFTSSTOR DFTSNAME      /* SET LINE TO ISRT */
      CALL INSERT_LINE 'AFTER', XL1, TLX
      NT = 'USING CLAUSE INSERTED WITH USER DEFAULT VALUES'
      CALL WRITE_MSG NT, (XL1+1)
      IF ( DFTSSTOR = 'STOGROUP' ) THEN DO
         TLX = MIN_DASD_ALLOC        /* SET LINE TO INSERT     */
         CALL INSERT_LINE 'AFTER', (XL1+1), TLX
         NT = 'MINIMUM PRIQTY AND SECQTY AMOUNTS INSERTED'
         CALL WRITE_MSG NT, (XL1+2)
         END
      /* DDL HAS BEEN MODIFIED BY EDIT MACRO.  SINCE NEW PARMS */
      ITERATE      /* HAVE BEEN INSERTED, RESTART DDL EDITING. */
      END          /* END OF USINGPOS = Ø,     NO USING CLAUSE */
   END  /* END OF SUB-SECTION USING CLAUSE FOR NON-PART TS.    */

/*-------------------------------------------------------------*/
/*  PARAGRAPH : PARSE/VALIDATE BUFFERPOOL CLAUSE IN CREATE TS. */
/*-------------------------------------------------------------*/
IF ( BPNMPOS > Ø ) THEN DO
   BPNAME = WORD(DDLSTR,(BPNMPOS+1))      /* GET BPNAME TOKEN  */
   CALL CHECK_BUFFERPOOL_PARM BPNAME      /* VALIDATE BPNAME   */
   IF ( BPNAME \= DFTSBPNM ) THEN DO
      'ISREDIT CURSOR = ' CL1 Ø
      'ISREDIT FIND BUFFERPOOL NEXT'
      'ISREDIT FIND "'BPNAME'" NEXT WORD'
      'ISREDIT (XL1,XC1) = CURSOR'
      IF ( SETDFLAG = YVAL ) THEN DO      /* CHANGE TO DEFAULT */
         'ISREDIT (LNX) = LINE' XL1
         PARSE VAR LNX F_TXT (BPNAME) E_TXT
         LNX = F_TXT DFTSBPNM
         CALL INSERT_LINE 'SAME', XL1, LNX
         NT = 'BPNAME CHANGED TO USER DEFAULT :' DFTSBPNM
         CALL WRITE_MSG NT, XL1
         END
```

```
      ELSE DO
         NT = 'USING BPNAME' BPNAME ', NOT DEFAULT' DFTSBPNM
         CALL WRITE_NOTE NT, XL1
         END                              /* END IF SETDFLAG ? */
      END                                 /* END IF BPNAME ?   */
   END                                    /* END IF BPNMPOS ?  */
ELSE DO                  /* NO BPNAME WAS FOUND, SET DEFAULTS */
   'ISREDIT CURSOR = ' CL1 Ø             /* REPOSITION CURSOR  */
   'ISREDIT FIND X"'SC_HEX_VAL'" NEXT'   /* FIND NEXT ; TOKEN  */
   'ISREDIT (XL1,XC1) = CURSOR'          /* GET CURSOR POS     */
   'ISREDIT TSPLIT' XL1 XC1-1            /* SPLIT LINE         */
   TLX = 'BUFFERPOOL 'DFTSBPNM           /* SET NEW LINE CONT. */
   CALL INSERT_LINE 'AFTER', XL1, TLX
   NT = DFTSBPNM 'USER DEFAULT BUFFERPOOL WILL BE USED'
   CALL WRITE_MSG NT, (XL1+1)            /* WRITE NOTE         */
   /*    DDL HAS BEEN MODIFIED BY EDIT MACRO.  SINCE NEW PARMS */
   ITERATE         /* HAVE BEEN INSERTED, RESTART DDL EDITING. */
   END                   /* END OF PARAGRAPH FOR BPNMPOS > Ø  */

/*-------------------------------------------------------------*/
/*  SUB-SECTION : PARSE/VALIDATE DDL FOR PARTITIONED T-SPACES  */
/*-------------------------------------------------------------*/
/*  USE VARIABLES SPLITRW1, SPLITCL1, SPLITRW2 AND SPLITCL2 TO */
/*  CONTROL POSITIONING THE CURSOR ON THE DDL LINES TO BE      */
/*  DELETED AND REPLACED BY THIS EDIT MACRO.                   */
/*-------------------------------------------------------------*/
IF ( NPRTPOS > Ø ) THEN DO          /* CHECK IT IS PARTITIONED */
   CALL INIT_VARS 'TS'              /* RESET VARS AND ARRAYS  */
   SPLITRW1 = Ø ; SPLITCL1 = Ø      /* INITIALIZE CURSOR VARS */
   SPLITRW2 = Ø ; SPLITCL2 = Ø      /* INITIALIZE CURSOR VARS */
   CALL PARSE_TABLESPACE_PARTS      /* CALL SUB-ROUTINE       */
   /* PARAGRAPH :  CALL COMPSIZE ROUTINE TO SIZE PARTS.       */
   DO PARTNO=1 TO NUMPARTS BY 1
      CALL COMPSIZE BPNAME, NUMPARTS, Ø, PARTNO
      END
   IF (PARM1 = 'SHOW') THEN CALL PRNTPART NUMPARTS
   /* PARAGRAPH :  CALL COMPSIZE ROUTINE TO SIZE PARTS.       */
   CALL UPDATE_TABLESPACE_PARTS_DDL  /* CALL SUB-ROUTINE      */
   END  /* END OF SUB-SECTION PARSE/VALIDATE DDL FOR P-TSPACES */

/*-------------------------------------------------------------*/
/*  SUB-SECTION : PROCESS SIMPLE OR SEGMENTED TABLESPACES.     */
/*  IF NON-PARTITIONED AND SEGSZ NOT SPECIFIED, SET IT TO ZERO */
/*-------------------------------------------------------------*/
IF ( NPRTPOS = Ø ) THEN DO          /* NON-PARTITIONED TSPACE. */
   /* PARAGRAPH : PARSE/EDIT/VALIDATE SEGSIZE CLAUSE.         */
   IF ( SEGMPOS > Ø ) THEN DO         /* IT IS A SEGMENTED TS */
      CALL CHECK_SEGSIZE_PARM         /* CALL SUB-ROUTINE     */
      P_SEGSIZE    = SEGSZ            /* SET ARRAY VALUE      */
      P_N_SEGSIZE  = SEGSZ            /* PREINITIALIZE        */
```

```
   END                                /* END FOR SEGMPOS > Ø  */
ELSE DO                               /* NON-SEGMENTED TS.    */
   SEGSZ = Ø                          /* SEGSIZE = Ø          */
   P_SEGSIZE    = Ø                   /* SEGSIZE = Ø          */
   P_N_SEGSIZE  = Ø                   /* PREINITIALIZE        */
   END                                /* END OF ELSE SEGMPOS  */
/*-------------------------------------------------------------*/
/*  PARAGRAPH : PARSE/EDIT PRIQTY CLAUSE IN CREATE TS. STMT.*/
/*  SEARCH FOR PRIQTY CLAUSE  IN THE DDL. IF FOUND, THEN    */
/*  PARSE THE PRIQTY VALUE AND CHECK IF IT'S A VALID NUMBER */
/*-------------------------------------------------------------*/
PRIQTYPOS = WORDPOS('PRIQTY',DDLSTR)
IF ( PRIQTYPOS > Ø ) THEN DO
   PRIQTY   = WORD(DDLSTR,(PRIQTYPOS+1))
   IF ( DATATYPE(PRIQTY,'NUM') \= 1 ) THEN DO
      LM = 'DDL ERROR # 3: PRIQTY VALUE NOT NUMERIC'
      SM = 'DDL ERROR # 3'
      CALL DDLERROR STOPFLAG, CL1, 'PRIQTY', PRIQTY, 1
      END
   END
ELSE DO                       /* USE DB2 DEFAULTS FOR MINIMUM */
   IF ( SUBSTR(BPNAME,1,5) \= 'BP32K' ) THEN ,
        PRIQTY = 12           /* SET MIN # OF PAGES FOR 4K TS */
   ELSE PRIQTY = 96           /* SET MIN # OF PAGES FOR 32KTS */
   END                        /* END OF SETTING DEFAULTS      */
P_PRIQTY.1  = PRIQTY          /* STORE VALUE IN WORK ARRAY    */

/*-------------------------------------------------------------*/
/*  PARAGRAPH : PARSE/EDIT SECQTY CLAUSE IN CREATE TS. STMT */
/*  SEARCH FOR SECQTY CLAUSE IN THE DDL. IF FOUND, THEN     */
/*  PARSE THE SECQTY VALUE AND CHECK IF IT'S A VALID NUMBER */
/*-------------------------------------------------------------*/
SECQTYPOS = WORDPOS('SECQTY',DDLSTR)
IF ( SECQTYPOS > Ø ) THEN DO
   SECQTY  = WORD(DDLSTR,(SECQTYPOS+1))
   IF ( DATATYPE(SECQTY,'NUM') \= 1 ) THEN DO
      LM = 'DDL ERROR # 4: SECQTY VALUE NOT NUMERIC'
      SM = 'DDL ERROR # 4'
      CALL DDLERROR STOPFLAG, CL1, 'SECQTY', SECQTY, 1
      END
   END                            /* END IF SECQTYPOS > Ø  */
ELSE DO             /* NO SECONDARY QUANTITY WAS SPECIFIED  */
   TQTY = PRIQTY % 1Ø              /* COMPUTE 1Ø% OF PRIQTY */
   IF ( SUBSTR(BPNAME,1,5) \= 'BP32K' ) THEN DO
      IF   ( TQTY > 12 ) THEN SECQTY = TQTY
      ELSE                    SECQTY = 12
      END                          /* END FOR 4K BUFFERPOOL */
   ELSE DO                         /* 32K BUFFERPOOLS       */
      IF   ( TQTY > 96 ) THEN SECQTY = TQTY
      ELSE                    SECQTY = 96
```

```
            END                          /* END FOR 32K BUFFERPOL */
          END             /* END OF ELSE SECQTYPOS > Ø. PARAGRAPH  */
       P_SECQTY.1  = SECQTY               /* STORE VALUE IN ARRAY  */

       /*-------------------------------------------------------------*/
       /*  PARAGRAPH : CALL COMPSIZE ROUTINE FOR SIZES                */
       /*-------------------------------------------------------------*/
       CALL COMPSIZE BPNAME, Ø, SEGSZ, 1
       IF (PARM1 = 'SHOW') THEN CALL PRNTPART 1

       /*-------------------------------------------------------------*/
       /*  PARAGRAPH : CALL UPDATE_TABLESPACE_DDL (NON-PARTITION)  */
       /*-------------------------------------------------------------*/
       CALL UPDATE_TABLESPACE_DDL 'CREATE'
       END  /* END PARAGRAPH   PARSE/VALIDATE DDL FOR NON-PART TS. */
    END       /* END OF CREATE TABLESPACE SECTION.                 */

/*****************************************************************/
/*  SECTION : PROCESS 'CREATE INDEX' STATEMENTS.                 */
/*****************************************************************/
/*  THE TEMPLATES FOR THE DIFFERENT CREATE INDEX STATEMENTS THAT  */
/*  WILL BE INTERPRETED BY THIS REXX EDIT MACRO ARE AS FOLLOWS :  */
/*---------------------------------------------------------------*/
/* CREATE               INDEX                          STATEMENT */
/* CREATE        UNIQUE INDEX                          STATEMENT */
/* CREATE TYPE X        INDEX                          STATEMENT */
/* CREATE TYPE X UNIQUE INDEX                          STATEMENT */
/* CREATE        UNIQUE WHERE NOT NULL INDEX           STATEMENT */
/* CREATE TYPE X UNIQUE WHERE NOT NULL INDEX           STATEMENT */
/*****************************************************************/
IF ( WORDPOS('CREATE',DDLSTR) = 1 ) & ,                 /* AND    */
   ( ( ( WORDPOS('INDEX',DDLSTR)  = 2 ) ) | ,           /* OR     */
     ( ( WORDPOS('UNIQUE',DDLSTR) = 2 )   & ,
       ( WORDPOS('INDEX',DDLSTR)  = 3 ) ) | ,           /* OR     */
     ( ( WORDPOS('TYPE',DDLSTR)   = 2 )   & ,
       ( WORDPOS('INDEX',DDLSTR)  = 4 ) ) | ,           /* OR     */
     ( ( WORDPOS('TYPE',DDLSTR)   = 2 )   & ,
       ( WORDPOS('UNIQUE',DDLSTR) = 4 )   & ,
       ( WORDPOS('INDEX',DDLSTR)  = 5 ) ) | ,           /* OR     */
     ( ( WORDPOS('UNIQUE',DDLSTR) = 2 )   & ,
       ( WORDPOS('WHERE',DDLSTR)  = 3 )   & ,
       ( WORDPOS('NOT',DDLSTR)    = 4 )   & ,
       ( WORDPOS('NULL',DDLSTR)   = 5 )   & ,
       ( WORDPOS('INDEX',DDLSTR)  = 6 ) ) | ,           /* OR     */
     ( ( WORDPOS('TYPE',DDLSTR)   = 2 )   & ,
       ( WORDPOS('UNIQUE',DDLSTR) = 4 )   & ,
       ( WORDPOS('WHERE',DDLSTR)  = 5 )   & ,
       ( WORDPOS('NOT',DDLSTR)    = 6 )   & ,
       ( WORDPOS('NULL',DDLSTR)   = 7 )   & ,
       ( WORDPOS('INDEX',DDLSTR)  = 8 ) ) ) THEN DO
```

```
/*-------------------------------------------------------------*/
/*  PARAGRAPH : DISPLAY SCREEN. SPLIT TOKENS & COUNT WORDS.    */
/*-------------------------------------------------------------*/
CALL DISPLAY_STATUS 'CREATE INDEX'
BADTK = 'NO'                        /* SET BAD TOKEN FLAG = NO  */
CALL SPLIT_DDL_TOKENS               /* SEPARATE TOKENS IN DDL   */
IF ( BADTK = YVAL ) THEN DO         /* CHECK IF BADTK WAS SET Y */
   'ISREDIT FIND CREATE WORD NEXT' /* FIND NEXT CREATE WORD    */
   LOOPCNTL = RC                    /* SET FIND RESULT/LOOPCNTL */
   ITERATE                          /* GOTO TOP OF THE DO WHILE */
   END
DDLWORDS = WORDS(DDLSTR)            /* COUNT THE NUMBER WORDS   */

/*-------------------------------------------------------------*/
/*  CHECK IF INDEX TYPE IS SPECIFIED. IF NOT, INSERT IT.       */
/*-------------------------------------------------------------*/
IF ( ( WORDPOS('TYPE',DDLSTR) = 2 ) & ,
     ( ( WORDPOS('1',DDLSTR) = 3 ) | ,
       ( WORDPOS('2',DDLSTR) = 3 ) ) ) THEN DO
   INDXTYPE = WORD(DDLSTR,3)                /* GET INDEX TYPE  */
   CALL CHECK_INDEXTYPE_PARM                /* VALIDATE PARM.  */
   IF ( INDXTYPE \= DFIXTYPE ) THEN DO      /* \= TO USR DFLT? */
      IF ( SETDFLAG = YVAL ) THEN DO        /* ENFORCE IS ON   */
         'ISREDIT CURSOR = ' CL1 Ø
         'ISREDIT FIND CREATE NEXT WORD'
         'ISREDIT FIND TYPE NEXT WORD'
         'ISREDIT FIND "'INDXTYPE'" NEXT WORD'
         'ISREDIT (XL1,XC1) = CURSOR'
         'ISREDIT (LNX) = LINE' XL1
         PARSE VAR LNX F_TEXT (INDXTYPE) E_TXT
         INDXTYPE = DFIXTYPE                /* RESET IDX TYPE  */
         LNX = (F_TEXT)(INDXTYPE)(E_TXT)    /* SET UPDATED DDL */
         CALL INSERT_LINE 'SAME', XL1, LNX  /* RE-WRITE DDL    */
         NT = 'INDEX TYPE CLAUSE CHANGED TO USER DEFAULT'
         CALL WRITE_MSG NT, XL1
         END
      ELSE DO
         'ISREDIT CURSOR = ' CL1 Ø
         'ISREDIT FIND CREATE NEXT WORD'
         'ISREDIT FIND TYPE NEXT WORD'
         'ISREDIT FIND "'INDXTYPE'" NEXT WORD'
         'ISREDIT (XL1,XC1) = CURSOR'
         NT = 'INDEX TYPE NOT THE SAME AS USER SPECIFIED'
         CALL WRITE_NOTE NT, XL1
         END
      END
   END                                      /* END OF IF ....  */
ELSE DO  /* INDEX TYPE IS NOT SPECIFIED.  INSERT IT INTO DDL   */
   INDXTYPE = DFIXTYPE                      /* SET INDEX TYPE  */
```

```
   'ISREDIT CURSOR = ' CL1 Ø
   'ISREDIT FIND "'WORD(DDLSTR,2)'" NEXT WORD'
   'ISREDIT (XL1,XC1) = CURSOR'
   'ISREDIT (LNX) = LINE' XL1
   PARSE VAR LNX 'CREATE ' E_TXT
   LNX = 'CREATE TYPE' DFIXTYPE E_TXT
   CALL INSERT_LINE 'SAME', XL1, LNX
   NT = 'INDEX TYPE CLAUSE WAS INSERTED INTO THE DDL'
   CALL WRITE_MSG NT, XL1
   /* SPLIT THE DDLSTR VARIABLE AND INSERT INDEX CLAUSE IN IT  */
   LPART = SUBWORD(DDLSTR,1,1)          /* SAVE LEFT OF DDLSTR */
   RPART = SUBWORD(DDLSTR,2)            /* SAVE RIGHT PART     */
   DDLSTR = LPART || ' TYPE ' DFIXTYPE || ' ' || RPART
   DDLWORDS = DDLWORDS + 2              /* INCREASE # OF WORDS */
   END                                      /* END OF ELSE ..  */

/*-------------------------------------------------------------*/
/*  PARAGRAPH : SEARCH FOR DIFFERENT CLAUSES IN THE DDL STMT.  */
/*-------------------------------------------------------------*/
BPNMPOS  = WORDPOS('BUFFERPOOL ',DDLSTR)    /* FIND BUFFERPOOL */
USINGPOS = WORDPOS('USING ',DDLSTR)         /* FIND USING      */
CLUSPOS  = WORDPOS('CLUSTER ',DDLSTR)       /* FIND CLUSTER    */
PARTPOS  = WORDPOS('PART ',DDLSTR)          /* FIND PART       */
NPRTPOS  = WORDPOS('NUMPARTS ',DDLSTR)      /* FIND NUMPARTS   */
SEGMPOS  = WORDPOS('SEGSIZE ',DDLSTR)       /* FIND SEGSIZE    */

/*-------------------------------------------------------------*/
/*  PARAGRAPH : ENSURE NUMPARTS CLAUSE IS NOT PRESENT.         */
/*-------------------------------------------------------------*/
IF ( NPRTPOS > Ø ) THEN DO          /* NUMPARTS IS FOR TABLESP */
   LM = 'DDL ERROR # 5: NUMPARTS CLAUSE NOT VALID FOR INDEXES'
   SM = 'DDL ERROR # 5'
   CALL DDLERROR STOPFLAG, CL1, 'NUMPARTS'
   END                             /* END IF NPRTPOS > Ø      */

/*-------------------------------------------------------------*/
/*  PARAGRAPH : ENSURE SEGSIZE  CLAUSE IS NOT PRESENT.         */
/*-------------------------------------------------------------*/
IF ( SEGMPOS > Ø ) THEN DO          /* NUMPARTS IS FOR TABLESP */
   LM = 'DDL ERROR # 6: SEGSIZE CLAUSE IS NOT VALID FOR INDEXES'
   SM = 'DDL ERROR # 6'
   CALL DDLERROR STOPFLAG, CL1, 'SEGSIZE'
   END                             /* END IF SEGMPOS > Ø      */

/*-------------------------------------------------------------*/
/*  PARAGRAPH : DETERMINE IF IT IS A CLUSTERED INDEX.  IF IT   */
/*  IS A CLUSTERED INDEX, DETERMINE IF IT IS A PARTITIONED ONE */
/*-------------------------------------------------------------*/
IF ( CLUSPOS > Ø ) THEN DO          /* CLUSTER MUST BE PRESENT */
   IF ( PARTPOS > Ø ) THEN DO      /* PART    MUST BE PRESENT */
```

```
      IF ( CLUSPOS < PARTPOS ) THEN PARTFLAG = 'YES'
      ELSE DO                       /* WRONG CLAUSE SEQUENCE   */
         LM = 'DDL ERROR # 7: WRONG POSITION FOR CLUSTER & PART'
         SM = 'DDL ERROR # 7'
         CALL DDLERROR YVAL, CL1, 'CLUSTER', 'PART', 1
         END                        /* END FOR ELSE DO...      */
      END                           /* END IF PARTPOS > Ø      */
   ELSE PARTFLAG = 'NO'             /* IT IS NOT PARTITIONED   */
   END                              /* END IF CLUSPOS > Ø      */
ELSE DO                    /* CHECK THAT PART IS NOT SPECIFIED */
   IF ( PARTPOS > Ø ) THEN DO
      LM = 'DDL ERROR # 8: PART CLAUSE USED, BUT NOT CLUSTER'
      SM = 'DDL ERROR # 8'
      CALL DDLERROR YVAL, CL1, 'PART'
      END                           /* END FOR ELSE DO...      */
   ELSE PARTFLAG = 'NO'             /* IT IS NOT PARTITIONED   */
   END                              /* END FOR ELSE DO...      */

/*---------------------------------------------------------------*/
/*  PARAGRAPH : SEARCH FOR DIFFERENT CLAUSES IN THE DDL STMT.  */
/*---------------------------------------------------------------*/
CALL CHECK_CLAUSES 'CREATE','INDEX'

/*---------------------------------------------------------------*/
/*  PARAGRAPH : PARSE/EDIT USING CLAUSE FOR PART. INDEXSPACES  */
/*---------------------------------------------------------------*/
/*  THIS PARAGRAPH WILL ENSURE THAT THERE IS AN EXPLICIT USING */
/*  CLAUSE IN THE CREATE INDEXSPACE STATEMENT BEFORE THE PARTS */
/*  CLAUSE.  THE USING CLAUSE FOR EACH INDEX PARTITION WILL BE */
/*  DETERMINED (PARSE/EDITED) ON LATER PARAGRAPHS.             */
/*---------------------------------------------------------------*/
/*  CHECK IF THE FIRST USING CLAUSE IS FOUND BEFORE OR AFTER   */
/*  THE PARTS CLAUSE, OR NOT FOUND AT ALL.  IF FOUND AND IT IS */
/*  AFTER THE PARTS CLAUSE OR NOT FOUND AT ALL, THEN INSERT AN */
/*  EXPLICIT USING CLAUSE USING THE VALUES SPECIFIED.          */
/*---------------------------------------------------------------*/
IF ( PARTFLAG = 'YES' ) THEN DO      /* IT IS PARTITIONED IX   */
   IF ( ( USINGPOS > CLUSPOS ) | ( USINGPOS = Ø ) ) THEN DO
      'ISREDIT CURSOR = ' CL1 Ø      /* POSITION CURSOR AT TOP */
      'ISREDIT FIND CLUSTER NEXT WORD'  /* FIND TKN TO SPLIT @ */
      'ISREDIT (XL1,XC1) = CURSOR'   /* GET CURRENT LINE POS   */
      'ISREDIT (LNX) = LINE' XL1     /* MAKE CONTENTS AVAILABLE*/
      PARSE VAR LNX F_TOKEN .        /* GET FIRST TOKEN IN LN  */
      IF \( F_TOKEN = 'CLUSTER' ) THEN DO
         'ISREDIT TSPLIT' XL1 XC1    /* NOT FIRST, SPLIT LINE  */
         XL1 = XL1 + 1               /* INCREASE LINE BY ONE   */
         END                         /* END OF NOT EQUAL       */
      /* INSERT THE USING CLAUSE BEFORE THE CLUSTER CLAUSE.    */
      TLX = 'USING ' DFIXSTOR DFIXNAME
      CALL INSERT_LINE 'BEFORE', XL1, TLX
```

```
      NT = 'USING CLAUSE INSERTED WITH USER DEFAULTS '
      CALL WRITE_MSG NT, XL1         /* WRITE NOTE BEFORE LINE */
      IF ( DFIXSTOR = 'STOGROUP' ) THEN DO
         TLX = MIN_DASD_ALLOC        /* SET LINE TO INSERT     */
         CALL INSERT_LINE 'AFTER', XL1, TLX
         NT = 'MINIMUM PRIQTY AND SECQTY AMOUNTS INSERTED'
         CALL WRITE_MSG NT, (XL1+1)
         END
      /* DDL HAS BEEN MODIFIED BY EDIT MACRO.  SINCE NEW PARMS */
      ITERATE      /* HAVE BEEN INSERTED, RESTART DDL EDITING. */
      END
   END  /* END OF PARTFLAG = 'YES'                              */

/*-------------------------------------------------------------*/
/* PARAGRAPH :  PARSE/EDIT USING CLAUSE NON-PARTITIONED INDEX. */
/* FIND A GOOD PLACE IN THE DDL TO INSERT THE USING CLAUSE.    */
/*-------------------------------------------------------------*/
IF ( PARTFLAG = 'NO' ) THEN DO       /* NOT PARTITIONED        */
   IF ( USINGPOS = Ø ) THEN DO       /* WE NEED TO FIND A SPOT */
      TKNPOS = Ø                     /* SET VARIABLE TO ZERO   */
      /* DO SEARCH FOR FREEPAGE TOKEN TO INSERT BEFORE         */
      IF ( TKNPOS = Ø ) THEN DO      /* IF NONE FOUND, SEEK    */
         TKWORD = 'FREEPAGE'         /* SET TEMP KEYWORD       */
         TKNPOS = WORDPOS(TKWORD,DDLSTR)      /* LOOK FOR IT   */
         END
      /* DO SEARCH FOR PCTFREE TOKEN TO INSERT BEFORE          */
      IF ( TKNPOS = Ø ) THEN DO      /* IF NONE FOUND, SEEK    */
         TKWORD = 'PCTFREE'          /* SET TEMP KEYWORD       */
         TKNPOS = WORDPOS(TKWORD,DDLSTR)      /* LOOK FOR IT   */
         END
      /* DO SEARCH FOR GBPCACHE BLOCK TO INSERT BEFORE         */
      IF ( TKNPOS = Ø ) THEN DO      /* IF NONE FOUND, SEEK    */
         TKWORD = 'GBPCACHE'         /* SET TEMP KEYWORD       */
         TKNPOS = WORDPOS(TKWORD,DDLSTR)      /* LOOK FOR IT   */
         END
      /* DO SEARCH FOR GBPCACHE BLOCK TO INSERT BEFORE         */
      IF ( TKNPOS = Ø ) THEN DO      /* IF NONE FOUND, SEEK    */
         TKWORD = 'SUBPAGES'         /* SET TEMP KEYWORD       */
         TKNPOS = WORDPOS(TKWORD,DDLSTR)      /* LOOK FOR IT   */
         END
      /* DO SEARCH FOR GBPCACHE BLOCK TO INSERT BEFORE         */
      IF ( TKNPOS = Ø ) THEN DO      /* IF NONE FOUND, SEEK    */
         TKWORD = 'BUFFERPOOL'       /* SET TEMP KEYWORD       */
         TKNPOS = WORDPOS(TKWORD,DDLSTR)      /* LOOK FOR IT   */
         END
      /* DO SEARCH FOR GBPCACHE BLOCK TO INSERT BEFORE         */
      IF ( TKNPOS = Ø ) THEN DO      /* IF NONE FOUND, SEEK    */
         TKWORD = 'CLOSE'            /* SET TEMP KEYWORD       */
         TKNPOS = WORDPOS(TKWORD,DDLSTR)      /* LOOK FOR IT   */
         END
```

```
/* DO SEARCH FOR GBPCACHE BLOCK TO INSERT BEFORE         */
IF ( TKNPOS = Ø ) THEN DO      /* IF NONE FOUND, SEEK    */
   TKWORD = 'DSETPASS'         /* SET TEMP KEYWORD       */
   TKNPOS = WORDPOS(TKWORD,DDLSTR)      /* LOOK FOR IT   */
   END
/* DO SEARCH FOR GBPCACHE BLOCK TO INSERT BEFORE         */
IF ( TKNPOS = Ø ) THEN DO      /* IF NONE FOUND, SEEK    */
   TKWORD = 'DEFER'            /* SET TEMP KEYWORD       */
   TKNPOS = WORDPOS(TKWORD,DDLSTR)      /* LOOK FOR IT   */
   END
/* DO SEARCH FOR GBPCACHE BLOCK TO INSERT BEFORE         */
IF ( TKNPOS = Ø ) THEN DO      /* IF NONE FOUND, SEEK    */
   TKWORD = 'PIECESIZE'        /* SET TEMP KEYWORD       */
   TKNPOS = WORDPOS(TKWORD,DDLSTR)      /* LOOK FOR IT   */
   END
/* SET SEMI-COLON TO BE THE TOKEN TO INSERT BEFORE.      */
IF ( TKNPOS = Ø ) THEN DO      /* IF NONE FOUND, SEEK    */
   TKWORD = ';'                /* SET TEMP KEYWORD       */
   TKNPOS = WORDPOS(TKWORD,DDLSTR)      /* LOOK FOR IT   */
   END
/* COULD NOT FIND A TOKEN.          ISSUE ERROR AND STOP  */
IF ( TKNPOS = Ø ) THEN DO      /* IF NONE, ISSUE ERROR   */
   LM = 'LOGIC ERROR 2: COULD NOT LOCATE PLACE TO INSERT'
   SM = 'LOGIC ERROR 2'
   CALL DDLERROR YVAL, CL1, 'CREATE', '' , 1
   END
/* NOW THAT WE FOUND A KEYWORD IN THE DDL, REPOSITION    */
/* THE CURSOR BEFORE THE TKWORD.  SAVE LINE NUMBER.      */
'ISREDIT CURSOR = ' CL1 Ø           /* REPOSITION CURSOR*/
IF ( TKWORD \= ';' ) THEN ,         /* NOT SEMI-COLON?  */
   'ISREDIT FIND "'TKWORD'" NEXT'  /* FIND TEMP KEYWORD*/
ELSE ,
   'ISREDIT FIND X"'SC_HEX_VAL'" NEXT' /* FIND NEXT SEMI */
/* SAVE CURSOR POSITION.                                 */
'ISREDIT (XL1,XC1) = CURSOR'   /* SAVE LINE AND COL POS. */
'ISREDIT (LNX) = LINE' XL1     /* MAKE CONTENTS AVAILABLE*/
PARSE VAR LNX F_TOKEN .        /* GET FIRST TOKEN IN LN  */
IF \( F_TOKEN = TKWORD ) THEN DO  /* IF FIRST TOKEN NOT  */
   'ISREDIT TSPLIT' XL1 (XC1-1)   /* THE SAME, SPLIT THE */
   XL1 = XL1 + 1               /* LINE AND INCREASE XL1  */
   END                         /* END OF NOT EQUAL       */
/* PROCEED TO INSERT THE USING CLAUSE BEFORE CURR LINE   */
TLX = 'USING ' DFIXSTOR DFIXNAME     /* SET LINE TO ISRT */
CALL INSERT_LINE 'BEFORE', XL1, TLX
NT = 'USING CLAUSE INSERTED WITH USER DEFAULT VALUES'
CALL WRITE_MSG NT, XL1
IF ( DFIXSTOR = 'STOGROUP' ) THEN DO
   TLX = MIN_DASD_ALLOC        /* SET LINE TO INSERT     */
   CALL INSERT_LINE 'AFTER', XL1, TLX
   NT = 'MINIMUM PRIQTY AND SECQTY AMOUNTS INSERTED'
```

```
         CALL WRITE_MSG NT, (XL1+1)
         END
      /* DDL HAS BEEN MODIFIED BY EDIT MACRO.  SINCE NEW PARMS */
      ITERATE      /* HAVE BEEN INSERTED, RESTART DDL EDITING. */
      END
   END                              /* END OF PARAGRAPH       */

/*-------------------------------------------------------------*/
/*  PARAGRAPH : PARSE/VALIDATE BUFFERPOOL CLAUSE.              */
/*-------------------------------------------------------------*/
IF ( BPNMPOS > Ø ) THEN DO
   BPNAME = WORD(DDLSTR,(BPNMPOS+1))
   CALL CHECK_BUFFERPOOL_PARM BPNAME      /* VALIDATE BPNAME   */
   IF ( BPNAME \= DFIXBPNM ) THEN DO
      'ISREDIT CURSOR = ' CL1 Ø
      'ISREDIT FIND BUFFERPOOL NEXT WORD'
      'ISREDIT FIND "'BPNAME'" NEXT WORD'
      'ISREDIT (XL1,XC1) = CURSOR'
      IF ( SETDFLAG = YVAL ) THEN DO      /* CHANGE TO DEFAULT */
         'ISREDIT (LNX) = LINE' XL1
         PARSE VAR LNX F_TXT BPNAME E_TXT
         LNX = F_TXT DFIXBPNM E_TXT
         CALL INSERT_LINE 'SAME', XL1, LNX
         NT = 'BPNAME CHANGED TO USER DEFAULT :' DFIXBPNM
         CALL WRITE_MSG NT, XL1
         END
      ELSE DO
         NT = 'USING BPNAME ' BPNAME ', NOT DEFAULT' DFIXBPNM
         CALL WRITE_NOTE NT, XL1
         END                              /* END SETDFLAG ?    */
      END                                 /* END BPNAME ?      */
   END               /* END OF IF BPNMPOS > Ø PARAGRAPH       */
ELSE DO              /* BPNMPOS = Ø, NO BUFFERPOOL FOUND.     */
   BPNAME = DFIXBPNM                     /* SET TO USER VALUE  */
   'ISREDIT CURSOR = ' CL1 Ø             /* REPOSITION CURSOR  */
   'ISREDIT FIND X"'SC_HEX_VAL'" NEXT'  /* FIND LAST DDL LINE  */
   'ISREDIT (XL1,XC1) = CURSOR'          /* GET CURSOR POS     */
   'ISREDIT TSPLIT' XL1 XC1-1            /* SPLIT LINE         */
   TLX = 'BUFFERPOOL 'BPNAME             /* SET NEW LINE CONT. */
   CALL INSERT_LINE 'AFTER', XL1, TLX
   NT = BPNAME ' BUFFERPOOL WILL BE USED'
   CALL WRITE_MSG NT, XL1                /* WRITE NOTE AFTER   */
   ITERATE           /* REPROCESS THE CURRENT DDL STATEMENT   */
   END               /* END OF ELSE BPNMPOS = Ø PARAGRAPH     */

/*-------------------------------------------------------------*/
/*  SUB-SECTION :  PARSE/EDIT/UPDATE DDL FOR PARTITIONED INDEX */
/*-------------------------------------------------------------*/
IF ( PARTFLAG = 'YES' ) THEN DO    /* IT IS A PARTITIONED IDX  */
   SPLITRW1 = Ø ; SPLITCL1 = Ø     /* INITIALIZE CURSOR VARS   */
```

```
   SPLITRW2 = Ø ; SPLITCL2 = Ø     /* INITIALIZE CURSOR VARS   */
   CALL INIT_VARS 'IX'                    /* INITIALIZE VARS   */
   HIGHEST_PARTNO = Ø                     /* INIT SPECIAL VAR  */
   CALL PARSE_INDEX_PARTS                 /* CALL SUB-ROUTINE  */
   NUMPARTS = HIGHEST_PARTNO              /* COPY SPECIAL VAR  */
   CALL CHECK_INDEX_PARTS                 /* CALL SUB-ROUTINE  */

   /*-------------------------------------------------------------*/
   /*  PARAGRAPH : CALL SUBROUTINE TO COMPUTE SIZES               */
   /*-------------------------------------------------------------*/
   DO PARTNO=1 TO NUMPARTS BY 1
      CALL COMPSIZE BPNAME, NUMPARTS, Ø, PARTNO
      END
   IF (PARM1 = 'SHOW') THEN CALL PRNTPART NUMPARTS

   /*-------------------------------------------------------------*/
   /*  PARAGRAPH : CALL SUBROUTINE TO UPDATE INDEX PART DDL.      */
   /*-------------------------------------------------------------*/
   CALL UPDATE_INDEX_PARTS_DDL
   END          /* END OF PROCESS CREATE INDEX SECTION           */

/*-----------------------------------------------------------------*/
/*  SUB-SECTION :  PARSE/EDIT/UPDATE DDL FOR NON-PART INDEX       */
/*-----------------------------------------------------------------*/
IF ( PARTFLAG = 'NO' ) THEN DO        /* NON-PARTITIONED IDX.  */
   /*-------------------------------------------------------------*/
   /*  PARAGRAPH :  PARSE/VALIDATE PRIQTY CLAUSE.                 */
   /*-------------------------------------------------------------*/
   PRIQTYPOS = WORDPOS('PRIQTY',DDLSTR)
   IF ( PRIQTYPOS > Ø ) THEN DO
      PRIQTY  = WORD(DDLSTR,(PRIQTYPOS+1))
      IF ( DATATYPE(PRIQTY,'NUM') \= 1 ) THEN DO
         LM = 'DDL ERROR # 1Ø: PRIQTY VALUE NOT NUMERIC'
         SM = 'DDL ERROR # 1Ø'
         CALL DDLERROR STOPFLAG, CL1, 'PRIQTY', PRIQTY, 1
         END
      END
   ELSE DO              /* NO PRIMARY   QUANTITY WAS SPECIFIED */
      IF ( SUBSTR(BPNAME,1,5) \= 'BP32K' ) THEN PRIQTY = 12
      ELSE PRIQTY = 96
      END
   P_PRIQTY.1  = PRIQTY               /* STORE VALUE IN ARRAY  */
   /*-------------------------------------------------------------*/
   /*  SEARCH FOR SECQTY CLAUSE  IN THE DDL. IF FOUND, THEN       */
   /*  PARSE SECQTY VALUE AND CHECK IT IS A NUMBER.               */
   /*-------------------------------------------------------------*/
   SECQTYPOS = WORDPOS('SECQTY',DDLSTR)
   IF ( SECQTYPOS > Ø ) THEN DO
      SECQTY  = WORD(DDLSTR,(SECQTYPOS+1))
      IF ( DATATYPE(SECQTY,'NUM') \= 1 ) THEN DO
```

```
            LM = 'DDL ERROR # 11: SECQTY VALUE NOT NUMERIC'
            SM = 'DDL ERROR # 11'
            CALL DDLERROR STOPFLAG, CL1, 'SECQTY', SECQTY, 1
            END
         END
      ELSE DO              /* NO SECONDARY QUANTITY WAS SPECIFIED */
         TQTY = PRIQTY % 10             /* COMPUTE 10% OF PRIQTY */
         IF ( SUBSTR(BPNAME,1,5) \= 'BP32K' ) THEN DO
            IF   ( TQTY > 12 ) THEN SECQTY = TQTY
            ELSE                    SECQTY = 12
            END
         ELSE DO                          /* 32K BUFFERPOOLS       */
            IF   ( TQTY > 96 )  THEN SECQTY = TQTY
            ELSE                     SECQTY = 96
            END
         END
      P_SECQTY.1  = SECQTY               /* STORE VALUE IN ARRAY  */
      /*-------------------------------------------------------------*/
      /*  CALL COMPSIZE FOR SPACE AND SEGMENT COMPUTATIONS.          */
      /*-------------------------------------------------------------*/
      CALL COMPSIZE BPNAME, 0, SEGSZ, 1
      IF (PARM1 = 'SHOW') THEN CALL PRNTPART 1
      /*-------------------------------------------------------------*/
      /*  CALL ROUTINE TO UPDATE THE DDL FOR NON-PARTITIONED INDEX*/
      /*-------------------------------------------------------------*/
      CALL UPDATE_INDEX_DDL 'CREATE'
      END                                /* END OF PARTFLAG = 'NO' */
   END             /* END OF SECTION CREATE INDEX                  */

/*****************************************************************/
/*  SECTION : PROCESS 'ALTER DATABASE' STATEMENT.                */
/*****************************************************************/
IF ( WORDPOS('ALTER',DDLSTR) = 1 ) & ,
   ( WORDPOS('DATABASE',DDLSTR) = 2 ) THEN DO

   /*-------------------------------------------------------------*/
   /*  PARAGRAPH : DISPLAY SCREEN. SPLIT TOKENS & COUNT WORDS.    */
   /*-------------------------------------------------------------*/
   CALL DISPLAY_STATUS 'ALTER DATABASE'
   BADTK = 'NO'                        /* SET BAD TOKEN FLAG = NO  */
   CALL SPLIT_DDL_TOKENS               /* SEPARATE TOKENS IN DDL   */
   IF ( BADTK = YVAL ) THEN DO         /* CHECK IF BADTK WAS SET Y */
      CALL FIND_NEXT_DDL_STMT          /* FIND NEXT VALID DDL STMT */
      ITERATE
      END
   DDLWORDS = WORDS(DDLSTR)           /* COUNT THE NUMBER OF WORDS */

   /*-------------------------------------------------------------*/
   /*  PARAGRAPH : CALL ROUTINE TO CHECK ALL CLAUSES IN ALTER     */
   /*-------------------------------------------------------------*/
```

```
CALL CHECK_CLAUSES 'ALTER','DATABASE'

/*-------------------------------------------------------------*/
/*  PARAGRAPH : PARSE/VALIDATE BUFFERPOOL IN ALTER DATABASE.   */
/*  IF CLAUSE IS MISSING, BYPASS CHECK.                        */
/*-------------------------------------------------------------*/
BPWORD = WORDPOS('BUFFERPOOL',DDLSTR)
IF ( BPWORD > Ø ) THEN DO
   PARSE VAR DDLSTR . 'BUFFERPOOL' DB_BPNAME .
   CALL CHECK_BUFFERPOOL_PARM DB_BPNAME
   BPNAME = STRIP(DB_BPNAME,B,';')
   IF ( BPNAME \= DFDBBPNM ) THEN DO
      'ISREDIT CURSOR = ' CL1 Ø
      'ISREDIT FIND BUFFERPOOL NEXT WORD'
      'ISREDIT FIND "'DB_BPNAME'" NEXT WORD'
      'ISREDIT (XL1,XC1) = CURSOR'
      IF ( SETDFLAG = YVAL ) THEN DO      /* CHANGE TO DEFAULT */
         'ISREDIT (LNX) = LINE' XL1
         PARSE VAR LNX F_TXT (DB_BPNAME) E_TXT
         LNX = F_TXT DFDBBPNM
         CALL INSERT_LINE 'SAME', XL1, LNX
         IF ( E_TXT \= '') THEN DO
            LNX = E_TXT
            CALL INSERT_LINE 'AFTER', XL1, LNX
            END
         NT = 'BPNAME NAME' BPNAME 'CHANGED TO DEFAULT' DFDBBPNM
         CALL WRITE_MSG NT, XL1
         END
      ELSE DO
         NT = BPNAME 'BUFFERPOOL NOT = TO USER DEFAULT' DFDBBPNM
         CALL WRITE_NOTE NT, XL1
         END                              /* END SETFLAG ?     */
      END                                 /* END DB_BPNAME ?   */
   END                                    /* END IF BPWORD > Ø */

/*-------------------------------------------------------------*/
/*  PARAGRAPH : PARSE/VALIDATE STOGROUP CLAUSE IN ALTER DB.    */
/*  IF CLAUSE IS MISSING, BYPASS CHECK.                        */
/*-------------------------------------------------------------*/
STOGPOS = WORDPOS('STOGROUP',DDLSTR)
IF ( STOGPOS \= Ø ) THEN DO          /* STOGROUP KEYWORD FOUND */
   STOGNAME = WORD(DDLSTR,(STOGPOS+1))
   STOGNAME = STRIP(STOGNAME,B,';')
   CALL CHECK_VALID_NAME STOGNAME, 'STOGROUP' /* VALIDATE NAME */
   IF ( STOGNAME \= DFDBSTOG ) THEN DO
      'ISREDIT CURSOR = ' CL1 Ø
      'ISREDIT FIND STOGROUP NEXT WORD'
      'ISREDIT FIND "'STOGNAME'" NEXT WORD'
      'ISREDIT (XL1,XC1) = CURSOR'
      IF ( SETDFLAG = YVAL ) THEN DO      /* CHANGE TO DEFAULT */
```

```
            'ISREDIT (LNX) = LINE' XL1
            /* CHECK TO SEE IF STOGROUP CLAUSE IS IN SAME LINE    */
            IF (WORDPOS('STOGROUP',LNX) > Ø ) THEN DO
               PARSE VAR LNX F_TXT 'STOGROUP' (STOGNAME) E_TXT
               LNX = F_TXT 'STOGROUP' DFDBSTOG
               END
            ELSE DO                 /* NO, IT IS NOT IN SAME LINE */
               PARSE VAR LNX F_TXT (STOGNAME) E_TXT
               LNX = F_TXT DFDBSTOG
               END
            CALL INSERT_LINE 'SAME', XL1, LNX
            IF ( E_TXT \= '' ) THEN DO
               LNX = E_TXT
               CALL INSERT_LINE 'AFTER', XL1, LNX
               END
            NT = 'STOGROUP NAME' STOGNAME,
                 'CHANGED TO DEFAULT' DFDBSTOG
            CALL WRITE_MSG NT, XL1
            END
         ELSE DO
            NT = 'USING STOGROUP' STOGNAME', NOT DEFAULT' DFDBSTOG
            CALL WRITE_NOTE NT, XL1
            END                                 /* IF SETDFLAG ?     */
         END                                    /* IF STOGNAME ?     */
      END                                       /* END IF STOGPOS\=Ø */
   END                                 /* END OF ALTER DB SECTION   */

/*****************************************************************/
/*  SECTION : PROCESS 'ALTER TABLESPACE' STATEMENTS.             */
/*****************************************************************/
IF ( ( WORDPOS('ALTER',DDLSTR)      = 1 ) & ,
     ( WORDPOS('TABLESPACE',DDLSTR) = 2 ) ) THEN DO

   /*-------------------------------------------------------------*/
   /*  PARAGRAPH : DISPLAY SCREEN. SPLIT TOKENS & COUNT WORDS.    */
   /*-------------------------------------------------------------*/
   CALL DISPLAY_STATUS 'ALTER TABLESPACE'
   BADTK = 'NO'                         /* SET BAD TOKEN FLAG = NO  */
   CALL SPLIT_DDL_TOKENS                /* SEPARATE TOKENS IN DDL   */
   IF ( BADTK = YVAL ) THEN DO          /* CHECK IF BADTK WAS SET Y */
      CALL FIND_NEXT_DDL_STMT           /* FIND NEXT VALID DDL STMT */
      ITERATE
      END
   DDLWORDS = WORDS(DDLSTR)             /* COUNT THE NUMBER OF WORDS */

   /*-------------------------------------------------------------*/
   /*  PARAGRAPH : CHECK THAT THE TABLESPACE NAME IS A VALID NAME */
   /*-------------------------------------------------------------*/
   TEMPNAME  = WORD(DDLSTR,3)     /* THE TSNAME IS THE THIRD WORD */
   PERIODPOS = POS('.',TEMPNAME)  /* IS THIS A 2 PART NAME ?      */
```

```
TEMPLEN   = LENGTH(TEMPNAME)   /* GET THE LENGTH OF THE STRING */
IF (PERIODPOS > Ø ) THEN DO    /* YES, IT IS A TWO PART NAME   */
   DBNAME = SUBSTR(TEMPNAME,1,(PERIODPOS-1))
   TSNAME = SUBSTR(TEMPNAME,(PERIODPOS+1),(TEMPLEN-PERIODPOS))
   CALL CHECK_VALID_NAME DBNAME, 'ALTER'
   CALL CHECK_VALID_NAME TSNAME, 'ALTER'
   END
ELSE DO                        /* TSNAME IS A ONE PART NAME    */
   CALL CHECK_VALID_NAME TEMPNAME, 'ALTER'
   'ISREDIT CURSOR = ' CL1 Ø
   'ISREDIT FIND (TEMPNAME) WORD NEXT'
   'ISREDIT (XL1,XC1) = CURSOR'
   NT = 'TABLESPACE 'TEMPNAME' WILL BE ASSUMED TO BE IN DSNDBØ4'
   CALL WRITE_NOTE NT, XL1
   END

/*-------------------------------------------------------------*/
/*  PARAGRAPH : INIT VARIABLES AND CHECK ALTER STMT CLAUSES.   */
/*-------------------------------------------------------------*/
CALL INIT_VARS 'TS'                  /* RESET VARS AND ARRAYS  */
CALL CHECK_CLAUSES 'ALTER','TABLESPACE'

/*-------------------------------------------------------------*/
/*  PARAGRAPH : PARSE/VALIDATE BUFFERPOOL CLAUSE IN ALTER  TS. */
/*-------------------------------------------------------------*/
BPNMPOS = WORDPOS('BUFFERPOOL',DDLSTR)    /* FIND BUFFERPOOL   */
IF ( BPNMPOS > Ø ) THEN DO
   BPNAME = WORD(DDLSTR,(BPNMPOS+1))      /* GET BPNAME TOKEN  */
   CALL CHECK_BUFFERPOOL_PARM BPNAME      /* VALIDATE BPNAME   */
   IF ( BPNAME \= DFTSBPNM ) THEN DO
      'ISREDIT CURSOR = ' CL1 Ø
      'ISREDIT FIND BUFFERPOOL NEXT WORD'
      'ISREDIT FIND "'BPNAME'" NEXT '
      'ISREDIT (XL1,XC1) = CURSOR'
      IF ( SETDFLAG = YVAL ) THEN DO      /* CHANGE TO DEFAULT */
         'ISREDIT (LNX) = LINE' XL1
         PARSE VAR LNX F_TXT (BPNAME) E_TXT
         LNX = F_TXT DFTSBPNM E_TXT
         CALL INSERT_LINE 'SAME', XL1, LNX
         NT = 'BPNAME CHANGED TO USER DEFAULT :' DFTSBPNM
         CALL WRITE_MSG NT, XL1
         END
      ELSE DO
         NT = 'USING BPNAME' BPNAME ', NOT DEFAULT'
         CALL WRITE_NOTE NT, XL1
         END                              /* END IF SETDFLAG ? */
      END                                 /* END IF BPNAME ?   */
   END                                    /* END IF BPNMPOS ?  */

/*-------------------------------------------------------------*/
```

```
   /*  PARAGRAPH: CHECK IF PRIQTY IS SPECIFIED.                    */
   /*---------------------------------------------------------------*/
   PRIQTYPOS = WORDPOS('PRIQTY',DDLSTR)
   IF ( PRIQTYPOS > Ø ) THEN DO
      PRIQTY   = WORD(DDLSTR,(PRIQTYPOS+1))
      IF ( DATATYPE(PRIQTY,'NUM') \= 1 ) THEN DO
         LM = 'DDL ERROR # 12: PRIQTY VALUE NOT NUMERIC'
         SM = 'DDL ERROR # 12'
         CALL DDLERROR STOPFLAG, CL1, 'PRIQTY', PRIQTY, 1
         END
      P_PRIQTY.1  = PRIQTY        /* STORE VALUE IN WORK ARRAY    */
      END

   /*---------------------------------------------------------------*/
   /*  PARAGRAPH: CHECK IF SECQTY IS SPECIFIED.                    */
   /*---------------------------------------------------------------*/
   SECQTYPOS = WORDPOS('SECQTY',DDLSTR)
   IF ( SECQTYPOS > Ø ) THEN DO
      SECQTY  = WORD(DDLSTR,(SECQTYPOS+1))
      IF ( DATATYPE(SECQTY,'NUM') \= 1 ) THEN DO
         LM = 'DDL ERROR # 13: SECQTY VALUE NOT NUMERIC'
         SM = 'DDL ERROR # 13'
         CALL DDLERROR STOPFLAG, CL1, 'SECQTY', SECQTY, 1
         END
      P_SECQTY.1  = SECQTY             /* STORE VALUE IN ARRAY    */
      END

   /*---------------------------------------------------------------*/
   /*  PARAGRAPH: BOTH PRIQTY AND SECQTY MUST BE FOUND FOR THIS   */
   /*  EDIT MACRO TO CALL THE COMPSIZE ROUTINE.  AN 'ALTER' PARM  */
   /*  IS PASSED TO THE UPDATE_TABLESPACE_DDL ROUTINE.            */
   /*---------------------------------------------------------------*/
   TRACE I
   IF ( ( PRIQTYPOS > Ø ) & ( SECQTYPOS > Ø ) ) THEN DO
      CALL COMPSIZE BPNAME, 1, Ø, 1
      CALL UPDATE_TABLESPACE_DDL 'ALTER'
      END
   TRACE OFF
   CALL PAUSEPRT
   END                        /* END OF ALTER TABLESPACE SECTION. */

/*****************************************************************/
/*  SECTION : PROCESS 'ALTER INDEX' STATEMENTS.                  */
/*****************************************************************/
IF ( ( WORDPOS('ALTER',DDLSTR) = 1 ) & ,
     ( WORDPOS('INDEX',DDLSTR) = 2 ) ) THEN DO

   /*---------------------------------------------------------------*/
   /*  PARAGRAPH : DISPLAY SCREEN. SPLIT TOKENS & COUNT WORDS.    */
   /*---------------------------------------------------------------*/
```

```
CALL DISPLAY_STATUS 'ALTER INDEX'
BADTK = 'NO'                          /* SET BAD TOKEN FLAG = NO  */
CALL SPLIT_DDL_TOKENS                 /* SEPARATE TOKENS IN DDL   */
IF ( BADTK = YVAL ) THEN DO           /* CHECK IF BADTK WAS SET Y */
   CALL FIND_NEXT_DDL_STMT            /* FIND NEXT VALID DDL STMT */
   ITERATE
   END
DDLWORDS = WORDS(DDLSTR)             /* COUNT THE NUMBER OF WORDS */

/*-------------------------------------------------------------*/
/*  CHECK IF INDEX TYPE IS SPECIFIED. IF NOT, WRITE A NOTE.    */
/*-------------------------------------------------------------*/
INDXTPOS = WORDPOS('TYPE',DDLSTR)           /* GET POSITION    */
IF ( INDXTPOS \= Ø ) THEN DO
   INDXTYPE = WORD(DDLSTR,(INDXTPOS+1))     /* GET INDEX TYPE  */
   CALL CHECK_INDEXTYPE_PARM                /* VALIDATE PARM.  */
   IF ( INDXTYPE \= DFIXTYPE ) THEN DO
      'ISREDIT CURSOR = ' CL1 Ø
      'ISREDIT FIND "'TYPE'" NEXT WORD'
      'ISREDIT (XL1,XC1) = CURSOR'
      NT = 'INDEX TYPE NOT THE SAME AS USER SPECIFIED'
      CALL WRITE_MSG NT, XL1
      END
   END                                      /* END OF IF ....  */

/*-------------------------------------------------------------*/
/*  PARAGRAPH : INIT VARIABLES AND CHECK ALTER STMT CLAUSES.   */
/*-------------------------------------------------------------*/
CALL INIT_VARS 'IX'                   /* RESET VARS AND ARRAYS  */
CALL CHECK_CLAUSES 'ALTER','INDEX'

/*-------------------------------------------------------------*/
/*  PARAGRAPH : PARSE/VALIDATE BUFFERPOOL CLAUSE IN ALTER  TS. */
/*-------------------------------------------------------------*/
BPNMPOS = WORDPOS('BUFFERPOOL',DDLSTR)    /* FIND BUFFERPOOL   */
IF ( BPNMPOS > Ø ) THEN DO
   BPNAME = WORD(DDLSTR,(BPNMPOS+1))      /* GET BPNAME TOKEN  */
   CALL CHECK_BUFFERPOOL_PARM BPNAME      /* VALIDATE BPNAME   */
   IF ( BPNAME \= DFTSBPNM ) THEN DO
      'ISREDIT CURSOR = ' CL1 Ø
      'ISREDIT FIND BUFFERPOOL NEXT WORD'
      'ISREDIT FIND "'BPNAME'" NEXT WORD'
      'ISREDIT (XL1,XC1) = CURSOR'
      IF ( SETDFLAG = YVAL ) THEN DO      /* CHANGE TO DEFAULT */
         'ISREDIT (LNX) = LINE' XL1
         PARSE VAR LNX F_TXT (BPNAME) E_TXT
         LNX = F_TXT DFIXBPNM E_TXT
         CALL INSERT_LINE 'SAME', XL1, LNX
         NT = 'BPNAME CHANGED TO USER DEFAULT :' DFIXBPNM
         CALL WRITE_MSG NT, XL1
```

```
            END
         ELSE DO
            NT = 'USING BPNAME' BPNAME ', NOT DEFAULT'
            CALL WRITE_NOTE NT, XL1
            END                                 /* END IF SETDFLAG ? */
         END                                    /* END IF BPNAME ?   */
      END                                       /* END IF BPNMPOS ?  */

   /*-------------------------------------------------------------*/
   /*  PARAGRAPH: CHECK IF PRIQTY IS SPECIFIED.                   */
   /*-------------------------------------------------------------*/
   PRIQTYPOS = WORDPOS('PRIQTY',DDLSTR)
   IF ( PRIQTYPOS > Ø ) THEN DO
      PRIQTY   = WORD(DDLSTR,(PRIQTYPOS+1))
      IF ( DATATYPE(PRIQTY,'NUM') \= 1 ) THEN DO
         LM = 'DDL ERROR # 14: PRIQTY VALUE NOT NUMERIC'
         SM = 'DDL ERROR # 14'
         CALL DDLERROR STOPFLAG, CL1, 'PRIQTY', PRIQTY, 1
         END
      P_PRIQTY.1  = PRIQTY        /* STORE VALUE IN WORK ARRAY    */
      END

   /*-------------------------------------------------------------*/
   /*  PARAGRAPH: CHECK IF SECQTY IS SPECIFIED.                   */
   /*-------------------------------------------------------------*/
   SECQTYPOS = WORDPOS('SECQTY',DDLSTR)
   IF ( SECQTYPOS > Ø ) THEN DO
      SECQTY  = WORD(DDLSTR,(SECQTYPOS+1))
      IF ( DATATYPE(SECQTY,'NUM') \= 1 ) THEN DO
         LM = 'DDL ERROR # 15: SECQTY VALUE NOT NUMERIC'
         SM = 'DDL ERROR # 15'
         CALL DDLERROR STOPFLAG, CL1, 'SECQTY', SECQTY, 1
         END
      P_SECQTY.1  = SECQTY            /* STORE VALUE IN ARRAY    */
      END

   /*-------------------------------------------------------------*/
   /*  PARAGRAPH: IF PRIQTY OR SECQTY WAS FOUND, PROCEED TO CALL  */
   /*  THE COMPSIZE AND UPDATE DDL ROUTINES.  'ALTER' PARAMETER   */
   /*  IS PASSED TO THE UPDATE_TABLESPACE_DDL ROUTINE SO IT WILL  */
   /*  BEHAVE DIFFERENTLY THAN WHEN PROCESSING A 'CREATE' STMT.   */
   /*-------------------------------------------------------------*/
   IF ( ( PRIQTYPOS > Ø ) & ( SECQTYPOS > Ø ) ) THEN DO
      CALL COMPSIZE BPNAME, 1, Ø, 1
      CALL UPDATE_INDEX_DDL 'ALTER'
      END
   END                        /* END OF ALTER TABLESPACE SECTION. */

/*----------------------------------------------------------------*/
/* PARAGRAPH :  FIND THE NEXT VALID DDL STATEMENT AND LOOP.       */
```

```
   /*-------------------------------------------------------------------*/
   CALL REPOSITION_CURSOR                 /* POSITION CURSOR ON ';' */
   CALL FIND_NEXT_DDL_STMT                /* FIND NEXT VALID DDL ST */
   /*-------------------------------------------------------------------*/
   /*   END OF MAIN SECTION - LOOP THRU ALL THE DDL STATEMENTS.     */
   /*-------------------------------------------------------------------*/
   END                                  /* END OF MAIN SECTION      */

/*---------------------------------------------------------------------*/
/*  FINAL PROCESS - DISPLAY MESSAGES AT TOP OF FILE.  REPOS CURSOR.  */
/*  ALSO, SET INUSE VARIABLE='NO' AND STORE IT IN THE PROFILE POOL.  */
/*---------------------------------------------------------------------*/
'ISREDIT (CURRLCNT) = LINENUM ' .ZLAST    /* GET CURRENT # OF LINES  */
NT = 'INITIAL NUMBER OF LINES IN FILE WERE : ' INITLCNT
CALL WRITE_NOTE NT, 1
NT = 'CURRENT NUMBER OF LINES IN FILE ARE  : ' CURRLCNT
CALL WRITE_NOTE NT, 1
INUSE = 'NO'                            /* SET INUSE='NO'.         */
'ISPEXEC VPUT ( INUSE ) PROFILE'        /* SAVE INTO PROFILE POOL. */
'ISREDIT UP MAX'                        /* GO TO TOP OF FILE       */
'ISREDIT LOCATE FIRST SPECIAL'          /* REPOSITION CURSOR       */

/*---------------------------------------------------------------------*/
/*  END OF MAIN EDIT MACRO CODE.  **  END OF MAIN EDIT MACRO CODE.   */
/*---------------------------------------------------------------------*/
EXIT                                /* EXIT OF MAIN CDB2DDL ROUTINE */

/*********************************************************************/
/*  SUB-ROUTINE FIND_NEXT_DDL_STMT: NO PARAMETERS PASSED.            */
/*  SUB-ROUTINE TO FIND THE NEXT ALTER OR CREATE STATEMENT IN THE    */
/*  DDL. IT WILL SET THE LOOPCNTL VARIABLE TO ZERO IF A VALID STMT   */
/*  WAS FOUND; IT WILL SET THE LOOPCNTL VARIABLE TO 1 IF NO MORE     */
/*  VALID STATEMENTS WERE FOUND.                                     */
/*********************************************************************/
FIND_NEXT_DDL_STMT: PROCEDURE EXPOSE CL1 CC1 CL2 CC2 YVAL STOPFLAG,
      TDSN TMEM EDITLINE LOOPCNTL DDLSTR SC_HEX_VAL PARM1
/*---------------------------------------------------------------------*/
/*  SUB-ROUTINE FIND_NEXT_DDL_STMT BEGINS HERE.                      */
/*---------------------------------------------------------------------*/
/*  SUB-ROUTINE LOGIG:  CHECK CONTENTS OF CURRENT LINE FOR VALID DDL */
/*  STATEMENT ( CREATE OR ALTER).  IF FOUND, THEN RETURN A ZERO.     */
/*---------------------------------------------------------------------*/
IF ( CL1 < CL2 ) THEN CL1 = CL2 + 1         /* REPOSITION CURSOR     */
LOOPCNTL = 1
'ISREDIT (NUMOFLNS) = LINENUM ' .ZLAST      /* GET CURRENT # OF LINES*/
DO WHILE ( ( LOOPCNTL = 1 ) & ( CL1 < NUMOFLNS ) )
   'ISREDIT CURSOR = ' CL1 Ø                /* SET CURSOR POSITION   */
   'ISREDIT (LNX) = LINE (CL1)'             /* GET CONTENTS OF LINE  */
   PARSE VAR LNX TOKEN .                    /* GET FIRST TOKEN       */
   CALL DISPLAY_STATUS 'IN SUB-ROUTINE FIND_NEXT_DDL_STMT'
```

```
   IF \ ( (TOKEN = 'CREATE') | (TOKEN = 'ALTER') ) THEN CL1 = CL1 + 1
   ELSE LOOPCNTL = Ø
   END
RETURN              /* END OF SUB-ROUTINE FIND_NEXT_DDL_STMT         */

/*********************************************************************/
/*  SUB-ROUTINE SPLIT_DDL_TOKENS: NO PARAMETERS PASSED.              */
/*  CRUCIAL ROUTINE BECAUSE IT WILL UPDATE BOTH THE DDLSTR VARIABLE  */
/*  AND THE DB2 DDL TEXT IN THE FILE BEING EDITED.  THE PURPOSE OF   */
/*  THIS ROUTINE IS TO ENSURE THAT KEY DELIMITERS SUCH AS COMMAS,    */
/*  PARENTHESES AND SEMICOLONS ARE SEPARATED FROM OTHER TOKENS.      */
/*  REFER TO SQL REFERENCE CONSTANTS SECTION FOR ADDITIONAL INFO.    */
/*********************************************************************/
SPLIT_DDL_TOKENS: PROCEDURE EXPOSE CL1 CL2 CC2 YVAL STOPFLAG,
                  BADTK DDLSTR SC_HEX_VAL PARM1
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE SPLIT_DDL_TOKENS BEGINS HERE.                        */
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE LOGIG:  TRAVERSE THE TEXT CONTAINED IN THE DDLSTR    */
/*  VARIABLE, AND SIMULTANEOUSLY, SET THE CURSOR IN THE FILE TEXT AS */
/*  THE DDL IS BEING TRAVERSED.  TO SEPARATE TWO TOKENS IN THE TEXT  */
/*  FILE, CAPTURE THE LINE, SPLIT THE TOKEN AND RE-WRITE THE LINE.   */
/*  TO SPLIT THE DDL STRING, SPLIT IT INTO TWO PARTS AND RE-JOIN     */
/*  THEM WITH THE NEW TOKEN IN-BETWEEN.                              */
/*-------------------------------------------------------------------*/
IF (PARM1 = 'SHOW' ) THEN CALL DISPLAY_DDLSTR     /* DISPLAY DDLSTR  */
STRING_FLAG = 'OFF'                          /* VAR TO TRACK STRINGS  */
STRING_DEL  = ''                             /* VAR TO TRACK STRINGS  */
'ISREDIT CURSOR = ' CL1 Ø                    /* SET CURSOR POSITION   */
CNTWORDS = WORDS(DDLSTR)                     /* SR LOCAL WORD COUNTER */
I = 1                                        /* SET BEGINNING VALUE   */
DO WHILE ( I <= CNTWORDS )                   /* LOOP THRU THE WORDS   */
   T2 = WORD(DDLSTR,I)                       /* GET WORD FROM LIST    */
   L2 = LENGTH(T2)                           /* GET WORD LENGTH       */
   IF ( T2 = ';' ) THEN ,                    /* IF TOKEN IS A SEMICLN */
        'ISREDIT FIND X"'SC_HEX_VAL'" NEXT' /* FIND NEXT SEMI-COLON  */
   ELSE 'ISREDIT FIND "'T2'" NEXT WORD'      /* ELSE FIND TOKEN WORD  */
   'ISREDIT (XL1,XC1) = CURSOR'              /* SAVE CURSOR POSITION  */
   /* IF A SINGLE CHAR, DETERMINE IF IT IS A STRING DELIMITER        */
   IF ( L2 = 1 ) THEN DO                     /* SINGLE CHARACTER      */
      IF ( ( T2 = '"' ) | ( T2 = "'" ) ) THEN DO   /* YES IT IS      */
         IF ( STRING_FLAG = 'OFF' ) THEN DO
            STRING_DEL  = T2                 /* SAVE STRING DELIMITER */
            STRING_FLAG = 'ON'               /* TURN STRING FLAG ON   */
            END
         ELSE DO                             /* STRING_FLAG = 'ON'    */
            IF ( STRING_DEL = T2 ) THEN DO   /* IS SAME DELIMITER ?   */
               STRING_DEL  = ''              /* CLEAR STRING DELIMIER */
               STRING_FLAG = 'OFF'           /* TURN STRING FLAG OFF  */
               END                           /* END OF DELIMITER CHK  */
```

```
         ELSE NOP                          /* LEAVE STRING_FLAG ON  */
         END                               /* END ELSE SECTION      */
      END                                  /* END OF '"' OR "'" CHK */
   I = I + 1                               /* INCREASE TOKEN PTR    */
   ITERATE                                 /* GO PROCESS NEXT TOKEN */
   END                                     /* END OF L = 1 IF STMT  */

/* MULTIPLE CHARACTERS.  CHECK IF WE HAVE A DELIMITER TOKEN.      */
IF ( ( POS('"',T2) = Ø ) & ( POS("'",T2 ) = Ø ) &,
     ( POS('(',T2) = Ø ) & ( POS(')',T2 ) = Ø ) &,
     ( POS(',',T2) = Ø ) & ( POS(';',T2 ) = Ø ) ) THEN DO
   I = I + 1                               /* INCREASE TOKEN PTR    */
   ITERATE                                 /* GO PROCESS NEXT TOKEN */
   END

/* WE HAVE A DELIMITER TOKEN.  DETERMINE WHAT TO DO WITH IT.      */
J = 1                                      /* SET CHAR POINTER      */
DO WHILE ( J <= L2 )                       /* LOOP THRU ALL CHARS   */
   CCHAR = SUBSTR(T2,J,1)                  /* GET CURRENT CHARACTER */
   /* IS THE CHARACTER A STRING DELIMITER ?                         */
   IF ( ( CCHAR = "'" ) | ( CCHAR = '"' ) ) THEN DO
      IF ( STRING_FLAG = 'OFF' ) THEN DO
         STRING_DEL  = CCHAR               /* SAVE STRING DELIMITER */
         STRING_FLAG = 'ON'                /* TURN STRING FLAG ON   */
         END
      ELSE DO                              /* STRING_FLAG = 'ON'    */
         /* IF STRING_DEL MATCHES CCHAR, IT IS POSSIBLE THAT IT     */
         /* IS THE END OF THE STRING PROVIDED THAT IT IS FOLLOWED */
         /* BY A CHARACTER NOT EQUAL TO ITSELF (CCHAR).           */
         IF ( STRING_DEL = CCHAR ) THEN DO   /* IS IT SAME DELI?  */
            IF ( J = L2 ) THEN DO
               STRING_DEL  = ''            /* CLEAR STRING DELIMIER */
               STRING_FLAG = 'OFF'         /* TURN STRING FLAG OFF  */
               END                         /* END OF DELIMITER CHK  */
         /* CHECK IF NEXT CHARACTER IS = TO THE CURRENT STRING    */
         /* DELIMITER. IF IT IS, THEN WE ARE NOT AT THE END OF    */
         /* THE CHARACTER STRING.  SEE SQL REFERENCE FOR DETAILS. */
            ELSE DO                          /* NOT @ LAST CHAR IN T2 */
               NCHAR = SUBSTR(T2,(J+1),1)    /* GET THE NEXT CHAR */
               IF ( NCHAR = STRING_DEL ) THEN NOP  /* LEAVE IT ON */
               ELSE DO
                  STRING_DEL  = ''           /* CLEAR STRING DELIMIER */
                  STRING_FLAG = 'OFF'        /* TURN STRING FLAG OFF  */
                  END                        /* ELSE NCHAR=STRING_DEL */
               END                           /* END ELSE SECTION      */
            END                              /* END STRING_DEL=CCHAR  */
         ELSE NOP                            /* LEAVE STRING_FLAG ON  */
         END                                 /* END ELSE SECTION      */
      J = J + 1                              /* INCREASE CHAR POINTER */
      ITERATE                                /* PROCESS NEXT CHAR     */
```

```
   END                              /* END STRING DELIMITER  */

/* IS THE CHARACTER ANY OF THE OTHER DELIMITERS ( PARENTHESES, */
/* SEMI-COLON OR COLON ) ?.  IF IT IS, AND THE EDIT MACRO IS   */
/* NOT CURRENTLY PROCESSING A STRING, THEN SPLIT THE CHARACTER */
IF ( ( CCHAR = '(' ) | ( CCHAR = ')' ) | ,          /* OR  */
     ( CCHAR = ";" ) | ( CCHAR = ',' ) ) &,         /* AND */
   ( STRING_FLAG = 'OFF' ) THEN DO
   /* SPLIT TOKEN INTO LEFT AND RIGTH CHARS. IF EITHER OF THEM */
   /* IS EMPTY, THEN WE WILL BE INSERTING ONLY 1 MORE TOKEN.   */
   /* IF BOTH ARE NOT EMPTY, THEN (DELTA) WE INSERT 2 TOKENS.  */
   /* SET T3 AND DELTA VARIABLES CORRESPONDINGLY.              */
   IF ( J = 1  ) THEN DO            /* CCHAR IS FIRST CHAR   */
      LCHARS = ''                   /* SET TO NIL.           */
      RCHARS = STRIP(SUBSTR(T2,2,(L2-1)))
      DELTA  = 1                    /* SET DELTA TO 1        */
      T3 = CCHAR RCHARS             /* JOIN TEXT FOR EDITING */
      END
   ELSE DO                          /* CCHAR NOT FIRST CHAR  */
      IF ( J = L2 ) THEN DO         /* CCHAR IS LAST CHAR    */
         LCHARS = STRIP(SUBSTR(T2,1,(L2-1)))
         RCHARS = ''                /* SET TO NIL.           */
         DELTA  = 1                 /* SET DELTA TO 1        */
         T3 = LCHARS CCHAR          /* JOIN TEXT FOR EDITING */
         END
      ELSE DO                       /* CCHAR IS IN MIDDLE    */
         LCHARS = STRIP(SUBSTR(T2,1,(J-1)))
         RCHARS = STRIP(SUBSTR(T2,(J+1),(L2-J)))
         DELTA  = 2                 /* SET DELTA TO 1        */
         T3 = LCHARS CCHAR RCHARS   /* JOIN TEXT FOR EDITING */
         END
      END                           /* END OF CCHAR \FIRST   */
   /* SPLIT THE DDLSTR VARIABLE, INSERT TOKENS, REJOIN DDLSTR  */
   LPART = SUBWORD(DDLSTR,1,(I-1))  /* SET LEFT PART OF LIST */
   RPART = SUBWORD(DDLSTR,(I+1))    /* SET RIGHT PART OF LST */
   DDLSTR = LPART LCHARS CCHAR RCHARS RPART
   /* INCREASE COUNTERS BY NUMBER OF TOKENS INSERTED IN DDLSTR */
   CNTWORDS = CNTWORDS + DELTA      /* INCREASE WORDS COUNT  */
   I  = I + 1                       /* INCREASE I COUNTER    */
   /* PROCESS THE TEXT IN THE FILE BEING EDITED.               */
   'ISREDIT (LNX) = LINE' XL1       /* GET CONTENTS OF LINE  */
   X_TEXT = STRIP(LNX)              /* EXTRACT THE TEXT ONLY */
   L_TEXT = LENGTH(X_TEXT)          /* GET LENGTH OF EX-TEXT */
   P_TEXT = POS(X_TEXT,LNX,1) - 1   /* SET # OF LEAD SPACES  */
   PARSE VAR X_TEXT F_TXT (T2) E_TXT  /* PARSE THE COMPONENTS  */
   IF ( L_TEXT > 7Ø ) THEN DO       /* CHECK EX-TEXT LENGTH  */
      LNX1 = COPIES(' ',P_TEXT)(F_TXT)       /* SPLIT LINE 1  */
      'ISREDIT LINE (XL1) = (LNX1)'          /* REWRITE LINE  */
      LNX2 = COPIES(' ',P_TEXT)(T3)(E_TXT)
      'ISREDIT LINE_AFTER (XL1) = (LNX2)'          /* INSERT  */
```

```
            END
         ELSE DO  /* TEXT IS NOT LONGER THAN 7Ø CHARS, ADD 1 SPACE   */
            LNX = COPIES(' ',P_TEXT)(F_TXT)(T3)(E_TXT)
            'ISREDIT LINE (XL1) = (LNX)'           /* REWRITE LINE  */
            END                                /* END OF ELSE > 7Ø...   */
         J = L2 + 1                            /* LEAVE J DO LOOP       */
         END                                   /* END OTHER DELIMITERS  */
      /* NOT A DELIMITER.  PROCESS NEXT CHAR IN TOKEN.                 */
      J = J + 1                                /* INCREASE J CHAR PTR   */
      ITERATE                                  /* DO NEXT CHARACTER     */
      END                                      /* END OF DO WHILE J<=L2 */
   I = I + 1                                   /* INCREASE I COUNTER    */
   END                                         /* END OF DO WHILE LOOP  */
/*-------------------------------------------------------------------*/
/*  REPOSITION CURSOR AND FIND NEW LOCATION OF ';' IN THE DDL STMT.  */
/*-------------------------------------------------------------------*/
'ISREDIT CURSOR = ' CL1 Ø                     /* SET CURSOR POSITION   */
'ISREDIT FIND X"'SC_HEX_VAL'" NEXT'           /* FIND NEXT ; TOKEN     */
'ISREDIT (CL2,CC2) = CURSOR'                  /* GET NEW CURSOR POS    */
/*-------------------------------------------------------------------*/
/*  CALL SUB-ROUTINE TO CHECK FOR PARENTHESES AND BAD TOKENS.        */
/*-------------------------------------------------------------------*/
IF (PARM1 = 'SHOW' ) THEN CALL DISPLAY_DDLSTR
CALL CHECK_FOR_BAD_TOKENS                     /* CALL SUB-ROUTINE      */
RETURN              /* END OF SUB-ROUTINE SPLIT_DDL_TOKENS           */

/*********************************************************************/
/*  SUB-ROUTINE CHECK_FOR_BAD_TOKENS: NO PARAMETERS PASSED.          */
/*********************************************************************/
CHECK_FOR_BAD_TOKENS: PROCEDURE EXPOSE CL1 CL2 CC2 YVAL STOPFLAG,
                      BADTK DDLSTR SC_HEX_VAL
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE CHECK_FOR_BAD_TOKENS BEGINS HERE.                    */
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE LOGIG:  TRAVERSE THE TEXT CONTAINED IN THE DDLSTR    */
/*  VARIABLE AND COUNT THE LEFT AND RIGHT PARENTHESES.  ALSO CHECK   */
/*  THAT ANY MINUS SIGNS OR AMPERSAND SIGNS ARE FOUND BETWEEN SINGLE */
/*  APOSTROPHES.  IF ANY ERRORS, ISSUE MESSAGES AND SET BADTK = 'Y'. */
/*-------------------------------------------------------------------*/
'ISREDIT CURSOR = ' CL1 Ø                     /* SET CURSOR POSITION   */
CNTWORDS = WORDS(DDLSTR)                      /* SR LOCAL WORD COUNTER */
I = 1                                         /* SET WORD COUNTER      */
CNT_RPAREN = Ø                                /* CNTR FOR RIGHT PARENT */
CNT_LPAREN = Ø                                /* CNTR FOR LEFT  PARENT */
STRING_FLAG = 'OFF'                           /* VAR TO TRACK STRINGS  */
STRING_DEL  = ''                              /* VAR TO TRACK STRINGS  */
DO WHILE ( I <= CNTWORDS )                    /* LOOP THRU THE WORDS   */
   T2 = WORD(DDLSTR,I)                        /* GET WORD FROM LIST    */
   L2 = LENGTH(T2)                            /* GET WORD LENGTH       */
   IF ( T2 = ';' ) THEN ,                     /* IF TOKEN IS A SEMICLN */
```

```
     'ISREDIT FIND X"'SC_HEX_VAL'" NEXT' /* FIND NEXT SEMI-COLON  */
ELSE 'ISREDIT FIND "'T2'" NEXT WORD'     /* FIND NEXT TOKEN WORD  */
'ISREDIT (XL1,XC1) = CURSOR'             /* GET  CURSOR POSITION  */

IF ( L2 = 1 ) THEN DO                    /* SINGLE CHAR STRING .  */
   IF ( ( T2 = "'" ) | ( T2 = '"') ) THEN DO /* STRING DELIMITER? */
      IF ( STRING_FLAG = 'OFF' ) THEN DO
         STRING_FLAG = 'ON'              /* TURN STRING FLAG ON   */
         STRING_DEL  = T2                /* SAVE STRING DELIMITER */
         END
      ELSE DO                            /* STRING FLAG IS ON     */
         IF ( STRING_DEL = T2 ) THEN DO  /* IS IT SAME DELIMITER? */
            STRING_FLAG = 'OFF'          /* TURN STRING FLAG OFF  */
            STRING_DEL  = ''             /* SET STRING_DEL TO NIL */
            END
         ELSE NOP
         END
      END
   IF ( ( T2 = '(' ) & ( STRING_FLAG = 'OFF' ) ) THEN,
      CNT_LPAREN = CNT_LPAREN + 1
   IF ( ( T2 = ')' ) & ( STRING_FLAG = 'OFF' ) ) THEN DO
      CNT_RPAREN = CNT_RPAREN + 1
      IF ( CNT_RPAREN > CNT_LPAREN ) THEN DO
         LM = 'DDL ERROR # 16: TOO MANY RIGHT PARENTHESES'
         SM = 'DDL ERROR # 16'
         CALL DDLERROR STOPFLAG, XL1, ')'
         BADTK = 'Y'
         LEAVE
         END
      END
   IF ( T2 = '-' ) THEN DO
      LM = 'DDL ERROR # 17: MINUS SIGN NOT ALLOWED IN THE DDL'
      SM = 'DDL ERROR # 17'
      CALL DDLERROR STOPFLAG, XL1, '-'
      BADTK = 'Y'
      LEAVE
      END
   IF ( T2 = '&' ) THEN DO
      LM = 'DDL ERROR # 18: AMPERSAND SIGN NOT ALLOWED IN THE DDL'
      SM = 'DDL ERROR # 18'
      CALL DDLERROR STOPFLAG, XL1, '&'
      BADTK = 'Y'
      LEAVE
      END
   I = I + 1                             /* INCREASE TOKEN PTR    */
   ITERATE                               /* GO PROCESS NEXT TOKEN */
   END                                   /* END IF LENGTH = 1     */

/* PROCESSING A MULTI-CHARACTER STRING TO COUNT FOR PARENTHESES   */
J = 1                                    /* SET CHAR POINTER      */
```

```
DO WHILE ( J <= L2 )                    /* LOOP THRU ALL CHARS   */
   CCHAR = SUBSTR(T2,J,1)               /* GET CURRENT CHARACTER */
   /* IS THE CHARACTER A STRING DELIMITER ?                      */
   IF ( ( CCHAR = "'" ) | ( CCHAR = '"' ) ) THEN DO
      IF ( STRING_FLAG = 'OFF' ) THEN DO
         STRING_DEL  = CCHAR            /* SAVE STRING DELIMITER */
         STRING_FLAG = 'ON'             /* TURN STRING FLAG ON   */
         END
      ELSE DO                           /* STRING_FLAG = 'ON'    */
         /* IF STRING_DEL MATCHES CCHAR, IT IS POSSIBLE THAT IT   */
         /* IS THE END OF THE STRING PROVIDED THAT IT IS FOLLOWED */
         /* BY A CHARACTER NOT EQUAL TO ITSELF (CCHAR).           */
         IF ( STRING_DEL = CCHAR ) THEN DO   /* IS IT SAME DELI?  */
            IF ( J = L2 ) THEN DO
               STRING_DEL  = ''         /* CLEAR STRING DELIMIER */
               STRING_FLAG = 'OFF'      /* TURN STRING FLAG OFF  */
               END                      /* END OF DELIMITER CHK  */
         /* CHECK IF NEXT CHARACTER IS = TO THE CURRENT STRING    */
         /* DELIMITER. IF IT IS, THEN WE ARE NOT AT THE END OF    */
         /* THE CHARACTER STRING.  SEE SQL REFERENCE FOR DETAILS. */
            ELSE DO                     /* NOT @ LAST CHAR IN T2 */
               NCHAR = SUBSTR(T2,(J+1),1)    /* GET THE NEXT CHAR */
               IF ( NCHAR = STRING_DEL ) THEN NOP  /* LEAVE IT ON */
               ELSE DO
                  STRING_DEL  = ''      /* CLEAR STRING DELIMIER */
                  STRING_FLAG = 'OFF'   /* TURN STRING FLAG OFF  */
                  END                   /* ELSE NCHAR=STRING_DEL */
               END                      /* END ELSE SECTION      */
            END                         /* END STRING_DEL=CCHAR  */
         END                            /* END ELSE SECTION      */
      J = J + 1                         /* INCREASE CHAR POINTER */
      ITERATE                           /* PROCESS NEXT CHAR     */
      END                               /* END STRING DELIMITER  */

   IF ( ( CCHAR = '(' ) & ( STRING_FLAG = 'OFF' ) ) THEN,
      CNT_LPAREN = CNT_LPAREN + 1
   IF ( ( CCHAR = ')' ) & ( STRING_FLAG = 'OFF' ) ) THEN DO
      CNT_RPAREN = CNT_RPAREN + 1
      IF ( CNT_RPAREN > CNT_LPAREN ) THEN DO
         LM = 'DDL ERROR # 19: TOO MANY RIGHT PARENTHESES'
         SM = 'DDL ERROR # 19'
         CALL DDLERROR STOPFLAG, XL1, ')'
         BADTK = 'Y'
         LEAVE
         END
      END
   IF ( ( CCHAR = '&' ) & ( STRING_FLAG = 'OFF' ) ) THEN DO
      LM = 'DDL ERROR # 20: AMPERSAND NOT ALLOWED IN THE DDL'
      SM = 'DDL ERROR # 20'
      CALL DDLERROR STOPFLAG, XL1, '&'
```

```
         BADTK = 'Y'
         LEAVE
         END
      IF ( ( CCHAR = '-' ) & ( STRING_FLAG = 'OFF' ) ) THEN DO
         LM = 'DDL ERROR # 21: MINUS SIGN NOT ALLOWED IN THE DDL'
         SM = 'DDL ERROR # 21'
         CALL DDLERROR STOPFLAG, XL1, '-'
         BADTK = 'Y'
         LEAVE
         END
      J = J + 1
      END                                    /* END OF DO WHILE J<=L2 */
   I = I + 1                                 /* INCREASE I COUNTER    */
   END                                       /* END OF DO WHILE LOOP  */

/* CHECK THAT THE PARENTHESES ARE BALANCED.                           */
IF ( CNT_RPAREN \= CNT_LPAREN ) THEN DO
   IF ( CNT_RPAREN < CNT_LPAREN ) THEN DO
      LM = 'DDL ERROR # 22: PARENTHESES NOT BALANCED. CHECK "("'
      SM = 'DDL ERROR # 22'
      CALL DDLERROR STOPFLAG, CL1, '('
      END
   ELSE DO
      LM = 'DDL ERROR # 23: PARENTHESES NOT BALANCED. CHECK ")"'
      SM = 'DDL ERROR # 23'
      CALL DDLERROR STOPFLAG, CL1, ')'
      END
   END

RETURN              /* END OF SUB-ROUTINE CHECK_FOR_BAD_TOKENS       */

/*********************************************************************/
/*  SUB-ROUTINE RESET_LAST_SPECIAL_MESSAGE                           */
/*********************************************************************/
RESET_LAST_SPECIAL_MESSAGE: PROCEDURE EXPOSE EDITLINE SC_HEX_VAL
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE DISPLAY_STATUS BEGINS HERE.                          */
/*-------------------------------------------------------------------*/
/*  THIS ROUTINE WILL RESET ALL SPECIAL MESSAGES THAT WERE CREATED   */
/*  **ON THE LAST** DDL STATEMENT EDITED BY THIS EDIT MACRO.         */
/*-------------------------------------------------------------------*/
IF ( EDITLINE > 1 ) THEN EDITLINE = EDITLINE - 1
'ISREDIT CURSOR = ' EDITLINE Ø              /* REPOSITION THE CURSOR */
'ISREDIT (TOPLN) = LINENUM ' .ZCSR          /* SET LABEL FOR TOPLN   */
'ISREDIT FIND X"'SC_HEX_VAL'" NEXT'         /* FIND NEXT ';' TOKEN   */
'ISREDIT (XLN1,XCC1) = CURSOR'              /* SAVE THE CURSOR POS   */
'ISREDIT (BOTTOMLN) = LINENUM ' .ZCSR       /* SET LABEL FOR BOTTOM  */
'ISREDIT RESET SPECIAL ' TOPLN BOTTOMLN     /* RESET ANY SPECIAL MSG */
RETURN                                      /* END OF RESET_LAST_... */
```

```
/*********************************************************************/
/*  SUB-ROUTINE DISPLAY_STATUS: PARMS TEXT1                          */
/*********************************************************************/
DISPLAY_STATUS: PROCEDURE EXPOSE CL1 DDLSTR TDSN TMEM
ARG TEXT1
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE DISPLAY_STATUS BEGINS HERE.                          */
/*-------------------------------------------------------------------*/
/*  THIS ROUTINE WILL DISPLAY A PANEL WITH INFORMATION OF WHAT IS    */
/*  BEING PROCESSED BY THE CDB2DDL EDIT MACRO.                       */
/*-------------------------------------------------------------------*/
ZEDLMSG = 'EDIT MACRO ERROR: REL_POS PARAMETER IS IN ERROR'
IF ( TMEM = '' ) THEN ,
     DDLMSG1 = 'EDITING FILE' TDSN
ELSE DDLMSG1 = 'EDITING FILE' TDSN || '(' || TMEM || ')'
DDLMSG2 = 'IN LINE ' CL1 ' -> ' TEXT1
IF ( LENGTH(DDLSTR) > 65 ) THEN DDLMSG3 = SUBSTR(DDLSTR,1,65)
ELSE                            DDLMSG3 = DDLSTR
'ISPEXEC CONTROL DISPLAY LOCK'              /* SET TO LOCK NEXT DISP */
'ISPEXEC DISPLAY PANEL(PDB2DDL3)'           /* DISPLAY PAN  PDB2DDL3 */
LRC = RC
IF ( LRC \= Ø ) THEN DO
   ZEDLMSG = 'EDIT MACRO ERROR: GOT RC 'LRC' IN DISPLAY_STATUS'
   ZEDSMSG = 'DISPLAY_STATUS ABEND'
   'ISPEXEC SETMSG MSG(ISRZØØ1)'
   EXIT 8                                    /* EXIT THE EDIT MACRO  */
   END
RETURN                                       /* END OF DISPLAY_STATUS */

/*********************************************************************/
/*  SUB-ROUTINE INSERT_LINE: PARMS REL_POS, LINENUM, NEWLINE         */
/*********************************************************************/
INSERT_LINE: PROCEDURE EXPOSE CL1 DDLSTR
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE INSERT_LINE BEGINS HERE.                             */
/*-------------------------------------------------------------------*/
/*  THIS ROUTINE WILL GET THE CONTENTS OF THE OLD LINE AND DETERMINE */
/*  IF IT CAN PLACE THE TEXT OF THE NEW LINE IN THE SAME POSITION AS */
/*  THE OLD ONE.                                                     */
/*-------------------------------------------------------------------*/
ARG REL_POS, LINENUM, NEWLINE
/*-------------------------------------------------------------------*/
/*  CHECK THE PARAMETERS PASSED TO THIS ROUTINE.                     */
/*-------------------------------------------------------------------*/
SC_HEX_VAL = C2X(';')   /* SET HEX VALUE OF SEMI-COLON TO SEARCH FOR */
IF \ ( ( REL_POS = 'SAME'  ) |,
       ( REL_POS = 'BEFORE') |,
       ( REL_POS = 'AFTER' ) ) THEN DO
   ZEDLMSG = 'EDIT MACRO ERROR: REL_POS PARAMETER IS IN ERROR'
   ZEDSMSG = 'INSERT_LINE ABEND'
```



```
   'ISPEXEC SETMSG MSG(ISRZØØ1)'
   EXIT 8                                    /* EXIT THE EDIT MACRO  */
   END
IF ( DATATYPE(LINENUM,'NUM') \= 1 ) THEN DO
   ZEDLMSG = 'EDIT MACRO ERROR: LINENUM IS NOT A VALID NUMBER'
   ZEDSMSG = 'INSERT_LINE ABEND'
   'ISPEXEC SETMSG MSG(ISRZØØ1)'
   EXIT 8                                   /* EXIT THE EDIT MACRO   */
   END                                      /* END OF IF NOT NUM     */
/*--------------------------------------------------------------------*/
/*  POSITION CURSOR AND GET CURRENT CONTENTS OF LINE POINTED TO BY    */
/*  THE LINENUM PARAMETER.  COPY CURRENT CONTENTS INTO TLN VARIABLE.  */
/*--------------------------------------------------------------------*/
'ISREDIT CURSOR = ' LINENUM Ø               /* SET CURSOR POSITION   */
LRC = RC                                    /* SET ISREDIT RETURN CD */
IF ( LRC \= Ø ) THEN SIGNAL ISR_ERROR       /* IF NOT ZERO, SIGNAL   */
'ISREDIT (TLN) = LINE' LINENUM              /* GET CURR-LN CONTENTS  */
LRC = RC                                    /* SET ISREDIT RETURN CD */
/*--------------------------------------------------------------------*/
/*  STRIP LEADING AND TRAILING BLANKS FROM THE TLN VARIABLE & STORE   */
/*  THE RESULT IN X_TLN. GET THE LENGTH INTO L_TLN.  USING THE X_TLN  */
/*  VARIABLE, FIND THE POSITION IN THE LINE WHERE X_TLN IS LOCATED.   */
/*--------------------------------------------------------------------*/
IF ( LRC \= Ø ) THEN P_TLN = 1              /* IF NOT ZERO, FAKE IT  */
ELSE DO                                     /* LINE FOUND, PROCESSIT */
   X_TLN  = STRIP(TLN)                      /* STRIP LEAD+TRAIL SPCS */
   L_TLN  = LENGTH(X_TLN)                   /* GET LENGTH OF TEXT    */
   P_TLN  = POS(X_TLN,TLN,1) - 1            /* FIND WHERE TEXT BEGIN */
   END
/*--------------------------------------------------------------------*/
/*  USE P_TLN AS THE INDENTATION COUNTER FOR THE OUTPUT LINE.         */
/*--------------------------------------------------------------------*/
X_NEWL = STRIP(NEWLINE)                     /* EXTRACT THE TEXT ONLY */
L_NEWL = LENGTH(X_NEWL)                     /* GET LENGTH OF EX-TEXT */
IF ( ( P_TLN + L_NEWL ) > 7Ø ) THEN P_NEWL = 7Ø - L_NEWL
ELSE                               P_NEWL = P_TLN
IF ( P_NEWL < 1 ) THEN P_NEWL = 1
NEWLINE = COPIES(' ',P_NEWL)(X_NEWL)
/*--------------------------------------------------------------------*/
/*  DEPENDING ON THE REL_POS PARAMETER, REPLACE OR INSERT NEWLINE     */
/*--------------------------------------------------------------------*/
IF ( REL_POS = 'SAME' ) THEN DO
   'ISREDIT LINE (LINENUM) = (NEWLINE)'            /* WRITE NEWLINE */
   LRC = RC
   END
IF ( REL_POS = 'BEFORE' ) THEN DO
   'ISREDIT LINE_BEFORE (LINENUM) = (NEWLINE)'     /* WRITE NEWLINE */
   LRC = RC
   END
IF ( REL_POS = 'AFTER' ) THEN DO
```

```
   'ISREDIT LINE_AFTER (LINENUM) = (NEWLINE)'       /* WRITE NEWLINE */
   LRC = RC
   END
/*-------------------------------------------------------------------*/
/*  PARAGRAPH TO DEAL WITH ISREDIT ERROR CONDITIONS.                 */
/*-------------------------------------------------------------------*/
ISR_ERROR: NOP
IF ( LRC \= Ø ) THEN DO
   ZEDLMSG = 'EDIT MACRO ERROR: GOT RETURN CODE 'LRC' IN INSERT_LINE'
   ZEDSMSG = 'INSERT_LINE ABEND'
   'ISPEXEC SETMSG MSG(ISRZØØ1)'
   EXIT 8                                    /* EXIT THE EDIT MACRO  */
   END
RETURN              /* END OF SUB-ROUTINE INSERT_LINE.               */

/*********************************************************************/
/*  SUB-ROUTINE INIT_VARS : ROUTINE TO INITIALIZE ARRAYS AND OTHER   */
/*  VARIABLES IN USE BY THE PROGRAM.  NO PARAMETERS ARE PASSED.      */
/*********************************************************************/
INIT_VARS: PROCEDURE EXPOSE P_COMPRESS. P_VALUES. ,
            DFTSSTOR     DFTSNAME     DFIXSTOR    DFIXNAME,
            P_STOR_TYPE. P_STOR_NAME. P_PRIQTY.   P_SECQTY. ,
            P_N_PRIQTY.  P_N_SECQTY.  P_SEGSIZE   P_N_SEGSIZE ,
            P_ERASE.     P_FREEPAGE.  P_PCTFREE.  P_GBPCACHE. ,
            CNTPART      CNTUSING     CNTPRIQTY   CNTSECQTY ,
            CNTCOMPRESS  CNTERASE     CNTFREEPAGE CNTPCTFREE ,
            CNTGBPCACHE  CNTVALUES
ARG OBJECT_TYPE
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE INIT_VARS BEGINS HERE.                               */
/*  INITIALIZE ALL ARRAY VARIABLES AND OTHER COUNTERS                */
/*-------------------------------------------------------------------*/
IF OBJECT_TYPE = 'TS' THEN DO
   P_STOR_TYPE. = DFTSSTOR                 /* SET TO USER DEFAULT   */
   P_STOR_NAME. = DFTSNAME                 /* SET TO USER DEFAULT   */
   END
IF OBJECT_TYPE = 'IX' THEN DO
   P_STOR_TYPE. = DFIXSTOR                 /* SET TO USER DEFAULT   */
   P_STOR_NAME. = DFIXNAME                 /* SET TO USER DEFAULT   */
   END
P_PRIQTY.    = Ø  ;      P_N_PRIQTY. = Ø    /* SET TO ZERO.          */
P_SECQTY.    = Ø  ;      P_N_SECQTY. = Ø    /* SET TO ZERO.          */
P_FREEPAGE.  = '' ;      P_PCTFREE.  = ''   /* SET DEFAULTS          */
P_GBPCACHE.  = '' ;      P_COMPRESS. = ''   /* SET DEFAULTS          */
P_VALUES.    = 'ERROR';  P_ERASE.    = 'NO' /* SET DEFAULTS          */
/* INITIALIZE SINGLE OCURRENCE VARIABLES FOR SEGMENTED TABLESPACES   */
P_SEGSIZE    = Ø  ;      P_N_SEGSIZE = Ø    /* SET TO ZEROS          */
/* INITIALIZE COUNTERS FOR TOKENS THAT CAN HAVE MULTIPLE OCCURRENCES */
CNTPART      = Ø ;       CNTUSING    = Ø    /* SET COUNTERS TO ZERO  */
CNTPRIQTY    = Ø ;       CNTSECQTY   = Ø    /* SET COUNTERS TO ZERO  */
```

```
CNTCOMPRESS  = Ø ;       CNTERASE    = Ø    /* SET COUNTERS TO ZERO  */
CNTFREEPAGE  = Ø ;       CNTPCTFREE  = Ø    /* SET COUNTERS TO ZERO  */
CNTVALUES    = Ø ;       CNTGBPCACHE = Ø    /* SET COUNTERS TO ZERO  */
RETURN                        /* END OF SUB-ROUTINE INIT_VARS       */

/*********************************************************************/
/*  SUB-ROUTINE CHECK_CLAUSES        ( FOR DB2/MVS V4 & V5 SYNTAX )  */
/*********************************************************************/
/*  THIS ROUTINE DOES : 1) COUNTS THE OCCURRENCES OF SEVERAL OF THE  */
/*  CLAUSES FOR THE CREATE/ALTER STATEMENTS FOR THE DATABASE, INDEX  */
/*  AND TABLESPACE DB2 OBJECTS.  IT CHECKS THAT EACH CLAUSE IS       */
/*  FOLLOWED BY THE APPROPIATE PARAMETER, AND IT ALSO CHECKS THAT    */
/*  THESE CLAUSES OCCURR THE APPROPIATE # OF TIMES.  2) IT ALSO      */
/*  CHECKS THAT EACH CLAUSE IS USED FOR THE APPROPIATE CREATE STMT.  */
/*  3) IT ALSO CHECKS THAT THE APPROPIATE CLAUSE IS USED FOR THE     */
/*  CORRECT VERSION OF DB2 SPECIFIED IN THE DEFAULTS PANEL.  4) IT   */
/*  CHECKS THAT THE APPROPIATE PARAMETER IS USED ON SEVERAL OF THE   */
/*  CLAUSES OF THE CREATE/ALTER STATEMENTS.                          */
/*********************************************************************/
/*  PARM VERSION  : DB2 VERSION SPECIFIED IN PANEL PDB2DDL1.         */
/*  PARM DDLFUNC  : TYPE OF DDL FUNCTION, ALTER OR CREATE.           */
/*  PARM OBJECT   : WHAT OBJECT TYPE IS REFERRED IN THE DDL.         */
/*  PARM DDLSTR   : VARIABLE CONTAINING THE DDL STRING.              */
/*  PARM CL1      : VARIABLE CONTAINING LINE WHERE DDLSTR BEGINS     */
/*********************************************************************/
CHECK_CLAUSES: PROCEDURE EXPOSE  CL1  DB2VER  DDLSTR  NUMPARTS,
          DFTSSTOR DFTSNAME DFIXSTOR DFIXNAME  SC_HEX_VAL YVAL,
          SETDFLAG STOPFLAG PARTFLAG INDXTYPE  HIGHEST_PARTNO
ARG DDLFUNC, OBJECT
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE CHECK_CLAUSES BEGINS HERE.                           */
/*-------------------------------------------------------------------*/
/*  PARAGRAPH : INITIALIZE LOCAL SUBROUTINE VARS TO COUNT OCCURRENCES*/
/*-------------------------------------------------------------------*/
CNT_ROSHARE   = Ø; CNT_STOGROUP = Ø;  CNT_WORKFILE  = Ø;   /* INIT   */
CNT_LARGE     = Ø; CNT_NUMPARTS = Ø;  CNT_SEGSIZE   = Ø;   /* INIT   */
CNT_BUFFERPOOL= Ø; CNT_LOCKSIZE = Ø;  CNT_LOCKMAX   = Ø;   /* INIT   */
CNT_CLOSE     = Ø; CNT_DSETPASS = Ø;  CNT_COMPRESS  = Ø;   /* INIT   */
CNT_CCSID     = Ø; CNT_LOCKPART = Ø;  CNT_MAXROWS   = Ø;   /* INIT   */
CNT_UNIQUE    = Ø; CNT_TYPE     = Ø;  CNT_PIECESIZE = Ø;   /* INIT   */
CNT_CLUSTER   = Ø; CNT_SUBPAGES = Ø;  CNT_DEFER     = Ø;   /* INIT   */
CNT_USING     = Ø; CNT_FREEPAGE = Ø;  CNT_PCTFREE   = Ø;   /* INIT   */
CNT_CONVERT   = Ø; CNT_GBPCACHE = Ø;  CNT_IN_DB     = Ø;   /* INIT   */
CNT_PRIQTY    = Ø; CNT_SECQTY   = Ø;  CNT_ERASE     = Ø;   /* INIT   */
CNT_VALUES    = Ø; CNT_ON_TABLE = Ø;  CNT_TABLESPACE= Ø;   /* INIT   */
CNT_PART      = Ø;                                         /* INIT   */

/*-------------------------------------------------------------------*/
/*  PARAGRAPH : INIT VARS TO KEEP PART NUMBER OF PREVIOUS OCCURRENCES*/
/*-------------------------------------------------------------------*/
```

```
PRE_USING     = Ø; PRE_PRIQTY   = Ø;  PRE_SECQTY    = Ø;   /* INIT   */
PRE_ERASE     = Ø; PRE_FREEPAGE = Ø;  PRE_PCTFREE   = Ø;   /* INIT   */
PRE_GBPCACHE  = Ø; PRE_COMPRESS = Ø;  PRE_VALUES    = Ø;   /* INIT   */

/*--------------------------------------------------------------------*/
/*  HIGHEST_PARTNO IS A SPECIAL VARIABLE USED TO KEEP THE RUNNING     */
/*  HIGHEST NUMBER OF A PART CLAUSE FOUND IN THE DDL AS IS BEING      */
/*  PARSED/CHECKED. THIS VARIABLE IS ALSO USED TO CONTROL WHEN TO     */
/*  START COUNTING THOSE CLAUSES THAT CAN BE FOUND MULTIPLE TIMES IN  */
/*  A PARTITIONED OBJECT, (INDEX OR TABLESPACE).                      */
/*--------------------------------------------------------------------*/
HIGHEST_PARTNO = Ø                     /* INIT SPECIAL VARIABLE     */
CURRENT_PARTNO = Ø                     /* INIT SPECIAL VARIABLE     */
/*--------------------------------------------------------------------*/
/*  MAX_COUNT IS A SPECIAL VARIABLE USED TO CONTROL HOW MANY TIMES    */
/*  A CLAUSE CAN BE FOUND IN A PARTITIONED OBJECT, DEPENDING ON THE   */
/*  POSITION WHERE THE CLOUSE IS FOUND.  FOR EXAMPLE, THE USING       */
/*  CLAUSE CAN BE FOUND BEFORE THE NUMPARTS OR PART CLAUSES ONLY      */
/*  ONCE, BUT IT CAN BE FOUND WITHIN THE PART SPECIFICATIONS UP TO    */
/*  THE HIGHEST PART NUMBER AVAILABLE.  SIMILARLY, AFTER THE PARTS    */
/*  SPECIFICATIONS, THE USING CLAUSE IS INVALID.                      */
/*--------------------------------------------------------------------*/
MAX_COUNT =      1                     /* INIT SPECIAL VARIABLE     */
/*--------------------------------------------------------------------*/
/*  PARAGRAPH : PARSE KEYWORDS AND COUNT THEIR OCCURRENCES.           */
/*--------------------------------------------------------------------*/
WORDCNT = WORDS(DDLSTR)                /* GET THE CNT_BER OF WORDS  */
LAST_GOOD_TOKEN = Ø                    /* SET LAST_GOOD_TOKEN VAR.  */
TOKEN = ''                             /* INITIALIZE TOKEN VARIABLE */
TCNT = Ø                               /* INITIALIZE TCNT TO ZERO   */
XL1 = CL1 ;  XC1 = Ø                   /* SET INITIAL POSITION VALS */
DO WHILE ( TCNT < WORDCNT )            /* LOOP THRU THE DDL TOKENS  */
   /* REPOSITION CURSOR IN CASE IT HAD BEEN MOVED BY SUBROUTINES     */
   'ISREDIT CURSOR = ' XL1 XC1         /* POS CURSOR AT CURR TOKEN  */
   IF ( TCNT <= LAST_GOOD_TOKEN ) THEN, /* IF TCNT < LAST_GOOD_TOKEN */
        TCNT = LAST_GOOD_TOKEN + 1     /* SET TCNT= LAST_GOOD_TOKEN */
   ELSE TCNT = TCNT + 1                /* INCREMENT TCNT POINTER    */
   PREV_TOKEN = TOKEN                  /* SAVE PREVIOUS TOKEN       */
   TOKEN = WORD(DDLSTR,TCNT)           /* MAKE LINE CONTENTS AVAIL  */

   IF ( TOKEN = ';' ) THEN ,               /* IF TOKEN IS A SEMICLN */
        'ISREDIT FIND X"'SC_HEX_VAL'" NEXT' /* FIND NEXT SEMI-COLON  */
   ELSE 'ISREDIT FIND "'TOKEN'" NEXT'      /* FIND NEXT TOKEN WORD  */
   'ISREDIT (XL1,XC1) = CURSOR'            /* GET  CURSOR POSITION  */

   /* PRELOAD THE NEXT TOKEN PAST THE CURRENT ONE FOR FUTURE PROCESS */
   IF (TCNT < WORDCNT) THEN N_TOKEN = WORD(DDLSTR,(TCNT+1))
   ELSE N_TOKEN = ''                   /* IF LAST WORD, SET TO NULL */
   /* PRELOAD THE SECOND NEXT TOKEN PAST THE CURRENT TOKEN.          */
   IF ((TCNT+1) < WORDCNT) THEN N2_TOKEN = WORD(DDLSTR,(TCNT+2))
```

```
   ELSE N2_TOKEN = ''                    /* IF LAST WORD, SET TO NULL */

/*-------------------------------------------------------------------*/
/*  PARAGRAPH : PERFORM CHECKS FOR EACH TOKEN FOUND.                 */
/*-------------------------------------------------------------------*/
   SELECT
      /*---------------------------------------------------------------*/
      /*   PROCESS SPECIAL SINGLE CHARACTER TOKENS FOUND IN THE DDL  */
      /*---------------------------------------------------------------*/
      WHEN TOKEN = ')' THEN DO
         IF ( DDLFUNC \= 'CREATE' ) THEN DO
            LM = 'DDL ERROR # 24: ")" NOT VALID IN ALTER STATEMENTS'
            SM = 'DDL ERROR # 24'
            CALL DDLERROR YVAL, XL1, ')'
            END
         /* IF LAST RIGHT PARENTHESES IN THE DDL, SET MAX_COUNT TO Ø */
         /* TO FORCE DDL ERRORS WHEN PART CLAUSES ARE FOUND PAST IT. */
         /* SET CURRENT_PARTNO TO 9999 TO CONTROL "TWICE" CHECKS.    */
         IF ( WORDPOS(')',DDLSTR,(TCNT+1)) = Ø ) THEN DO
            CURRENT_PARTNO = 9999  /* NO MORE PARTS WILL BE FOUND    */
            MAX_COUNT = Ø          /* SET COUNT LIMIT TO ZERO.       */
            END
         ITERATE
         END                              /* END WHEN TOKEN=')'       */
      WHEN TOKEN = '(' THEN DO
         IF ( DDLFUNC \= 'CREATE' ) THEN DO
            LM = 'DDL ERROR # 25: "(" NOT VALID IN ALTER STATEMENTS'
            SM = 'DDL ERROR # 25'
            CALL DDLERROR YVAL, XL1, '('
            END
         ITERATE
         END                              /* END WHEN TOKEN='('       */
      WHEN TOKEN = ',' THEN ITERATE
      WHEN TOKEN = ';' THEN ITERATE
      /*---------------------------------------------------------------*/
      /*   PROCESS NUMPARTS CLAUSE FOR PARTITIONED TABLESPACES.      */
      /*---------------------------------------------------------------*/
      WHEN TOKEN = 'NUMPARTS' THEN DO
         LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
         CNT_NUMPARTS = CNT_NUMPARTS + 1
         IF CNT_NUMPARTS > 1 THEN DO
            LM = 'DDL ERROR # 26: TOO MANY "NUMPARTS" CLAUSES'
            SM = 'DDL ERROR # 26'
            CALL DDLERROR YVAL, XL1, 'NUMPARTS'
            END                           /* END OF CNT_NUMPARTS > 1  */
         IF \((DDLFUNC='CREATE') & (OBJECT='TABLESPACE')) THEN DO
            LM = 'DDL ERROR # 27: USE NUMPARTS ONLY ON CREATE TS'
            SM = 'DDL ERROR # 27'
            CALL DDLERROR YVAL, XL1, 'NUMPARTS'
            END                           /* END OF IF STMT           */
```

```
   IF ( DATATYPE(N_TOKEN,'NUM') \= 1 ) THEN DO
      LM = 'DDL ERROR # 28: INVALID NUMPARTS PARM. MUST BE #'
      SM = 'DDL ERROR # 28'
      CALL DDLERROR YVAL, CL1, 'NUMPARTS', TOKEN
      END                          /* END OF DATATYPE CHECK.   */
   IF ( ( CNT_LARGE = Ø ) & ( N_TOKEN > 64 ) ) THEN DO
      LM = 'DDL ERROR # 29: NUMPARTS RANGE 1-64 FOR NON-LARGE TS'
      SM = 'DDL ERROR # 29'
      CALL DDLERROR YVAL, CL1, 'NUMPARTS', TOKEN
      END                          /* END OF DATATYPE CHECK.   */
   /* SET UP THE MAX_COUNT TO BE NUMPARTS.                      */
   MAX_COUNT = N_TOKEN
   /* RESET THE COUNTERS FOR ALL PARTITIONED CLAUSES            */
   CNT_USING = Ø    ; CNT_PRIQTY = Ø   ;  CNT_SECQTY = Ø
   CNT_ERASE = Ø    ; CNT_FREEPAGE = Ø ;  CNT_PCTFREE = Ø
   CNT_GBPCACHE = Ø ; CNT_COMPRESS = Ø
   PRE_USING = Ø    ; PRE_PRIQTY = Ø   ;  PRE_SECQTY = Ø
   PRE_ERASE = Ø    ; PRE_FREEPAGE = Ø ;  PRE_PCTFREE = Ø
   PRE_GBPCACHE = Ø ; PRE_COMPRESS = Ø
   END                          /* END TOKEN='NUMPARTS'     */
/*-------------------------------------------------------------*/
/*   GROUP LOGIC FOR TOKENS THAT MIGHT BE FOUND MORE THAN 1    */
/*-------------------------------------------------------------*/
WHEN TOKEN = 'PART' THEN DO
   CNT_PART = CNT_PART + 1
   IF ( PARTFLAG = 'NO' ) THEN DO
      LM = 'DDL ERROR # 3Ø: 'OBJECT' NOT PARTITIONED BUT FOUND',
           '"PART"'
      SM = 'DDL ERROR # 3Ø'
      CALL DDLERROR YVAL, XL1, 'PART'
      END                          /* END OF PARTFLAG = 'NO'   */
   IF ( DATATYPE(N_TOKEN,'NUM') \= 1 ) THEN DO
      LM = 'DDL ERROR # 31: INVALID PART NUMBER. ENTER NUMBER'
      SM = 'DDL ERROR # 31'
      CALL DDLERROR YVAL, XL1, 'PART', N_TOKEN
      END                          /* END OF DATATYPE CHECK    */
   CURRENT_PARTNO = N_TOKEN        /* SET CURRENT PART NUMBER  */
   LAST_GOOD_TOKEN = TCNT + 1      /* INC GOOD TOKEN POINTER   */
   /* KEEP TRACK OF HIGHEST INDEX PART NUMBER IN THE DDL.      */
   /* WHENEVER THE FIRST PART CLAUSE IS FOUND, RESET COUNTERS  */
   /* FOR THOSE CLAUSES THAT CAN BE FOUND BEFORE PART CLAUSES. */
   IF ( OBJECT = 'INDEX' ) THEN DO
      IF ( HIGHEST_PARTNO = Ø ) THEN DO   /* FIRST TIME AROUND */
         CNT_USING = Ø    ; CNT_PRIQTY = Ø   ;  CNT_SECQTY = Ø
         CNT_ERASE = Ø    ; CNT_FREEPAGE = Ø ;  CNT_PCTFREE = Ø
         CNT_GBPCACHE = Ø ; CNT_COMPRESS = Ø ;  CNT_VALUES  = Ø
         PRE_USING = Ø    ; PRE_PRIQTY = Ø   ;  PRE_SECQTY = Ø
         PRE_ERASE = Ø    ; PRE_FREEPAGE = Ø ;  PRE_PCTFREE = Ø
         PRE_GBPCACHE = Ø ; PRE_COMPRESS = Ø ;  PRE_VALUES  = Ø
         END                          /* END FIRST TIME    */
```

```
      IF ( N_TOKEN > HIGHEST_PARTNO ) THEN DO
         HIGHEST_PARTNO = N_TOKEN
         MAX_COUNT      = N_TOKEN
         END
      END                          /* END IF OBJECT = 'INDEX'  */
   END                             /* END WHEN TOKEN='PART'    */
WHEN TOKEN = 'VALUES' THEN DO
   /* CHECK CLAUSE NOT FOUND IN SAME PART DDL SECTION          */
   IF ( CURRENT_PARTNO > Ø ) THEN DO
      IF ( PRE_VALUES = CURRENT_PARTNO ) THEN DO
         LM = 'DDL ERROR # 32: "VALUES" CLAUSE FOUND TWICE'
         SM = 'DDL ERROR # 32'
         CALL DDLERROR YVAL, XL1, 'VALUES'
         END
      PRE_VALUES = CURRENT_PARTNO
      END
   NEXT_RPAREN = WORDPOS(')',DDLSTR,TCNT)  /* GET POS OF ')'   */
   LAST_GOOD_TOKEN = NEXT_RPAREN  /* SET VAR USED FOR CHECKS   */
   CNT_VALUES = CNT_VALUES + 1
   /* CHECK THAT VALUES IS ONLY SPECIFIED FOR PARTITIONED IDXS */
   IF ( ( DDLFUNC = 'CREATE' ) & ( OBJECT = 'INDEX' ) ) THEN DO
      IF ( PARTFLAG = 'NO' ) THEN DO
         LM = 'DDL ERROR # 33: "VALUES" CLAUSE NOT VALID FOR',
              ' NON-PARTITIONED INDEX'
         SM = 'DDL ERROR # 33'
         CALL DDLERROR YVAL, XL1, 'VALUES'
         END                       /* END OF PARTFLAG = 'NO'    */
      IF ( CNT_VALUES > MAX_COUNT ) THEN DO
         LM = 'DDL ERROR # 34: TOO MANY "VALUES" CLAUSE FOUND'
         SM = 'DDL ERROR # 34'
         CALL DDLERROR YVAL, XL1, 'VALUES'
         END                       /* END OF CNT_VALUES > HPN   */
      END                          /* END OF DDLFUNC='CREATE'   */
   ELSE DO
      LM = 'DDL ERROR # 35: "VALUES" CLAUSE NOT VALID FOR',
           DDLFUNC OBJECT
      SM = 'DDL ERROR # 35'
      CALL DDLERROR YVAL, XL1, 'VALUES'
      END
   END                             /* END WHEN TOKEN='VALUES'  */
WHEN TOKEN = 'USING' THEN DO
   /* CHECK CLAUSE NOT FOUND IN SAME PART DDL SECTION          */
   IF ( CURRENT_PARTNO > Ø ) THEN DO
      IF ( PRE_USING = CURRENT_PARTNO ) THEN DO
         LM = 'DDL ERROR # 36: "USING" CLAUSE FOUND TWICE'
         SM = 'DDL ERROR # 36'
         CALL DDLERROR YVAL, XL1, 'USING'
         END
      PRE_USING = CURRENT_PARTNO
      END
```

```
      LAST_GOOD_TOKEN = TCNT + 2     /* SET VAR USED FOR CHECKS   */
      CNT_USING = CNT_USING + 1
      /* ALLOW MULTIPLE USING CLAUSES FOR CLUSTER IX AND PART-TS  */
      IF ( DDLFUNC = 'CREATE' ) THEN DO
         IF ( ( OBJECT  = 'TABLESPACE' ) & ( NUMPARTS > Ø ) & ,
              ( CNT_USING > MAX_COUNT ) ) | ,
            ( ( OBJECT  = 'TABLESPACE' ) & ( NUMPARTS = Ø ) & ,
              ( CNT_USING > 1 ) ) |,
            ( ( OBJECT  = 'INDEX' )      & ( PARTFLAG = 'YES' ) &,
              ( CNT_USING > MAX_COUNT ) ) |,
            ( ( OBJECT  = 'INDEX' )      & ( PARTFLAG = 'NO'  ) &,
              ( CNT_USING > 1 ) ) THEN DO
            LM = 'DDL ERROR # 37: TOO MANY "USING" CLAUSES'
            SM = 'DDL ERROR # 37'
            CALL DDLERROR YVAL, XL1, 'USING'
            END                      /* END OF MULTIPLE CHECKS   */
         END                         /* END OF DDLFUNC='CREATE'  */
      ELSE DO                        /* DDLFUNC \= 'CREATE'      */
         IF ( CNT_USING > 1 ) THEN DO
            LM = 'DDL ERROR # 38: TOO MANY "USING" CLAUSES'
            SM = 'DDL ERROR # 38'
            CALL DDLERROR YVAL, XL1, 'USING'
            END                      /* END OF IF CNT_USING > 1  */
         END                         /* END OF ELSE DDLFUNC \=.. */
      IF \( N_TOKEN= 'VCAT' | N_TOKEN = 'STOGROUP' ) THEN DO
         LM = 'DDL ERROR # 39: INVALID USING PARAMETER. CHECK PARM'
         SM = 'DDL ERROR # 39'
         CALL DDLERROR YVAL, XL1, N_TOKEN
         END                         /* END OF CHECK N_TOKEN KWD */
      LAST_USING_TYPE = N_TOKEN      /* SAVE USING TYPE          */
      IF ( OBJECT = 'TABLESPACE' ) THEN,
         CALL CHECK_USING_PARM XL1, 'TS', N_TOKEN, N2_TOKEN
      IF ( OBJECT = 'INDEX'      ) THEN,
         CALL CHECK_USING_PARM XL1, 'IX', N_TOKEN, N2_TOKEN
      END                            /* END TOKEN='USING'        */
   WHEN TOKEN = 'PRIQTY' THEN DO
      /* CHECK CLAUSE NOT FOUND IN SAME PART DDL SECTION          */
      IF ( CURRENT_PARTNO > Ø ) THEN DO
         IF ( PRE_PRIQTY = CURRENT_PARTNO ) THEN DO
            LM = 'DDL ERROR # 4Ø: "PRIQTY" CLAUSE FOUND TWICE'
            SM = 'DDL ERROR # 4Ø'
            CALL DDLERROR YVAL, XL1, 'PRIQTY'
            END
         PRE_PRIQTY = CURRENT_PARTNO
         END
      LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
      CNT_PRIQTY = CNT_PRIQTY + 1
      /* ALLOW MULTIPLE PRIQTYS FOR CLUSTER IX AND PART-TS        */
      IF ( DDLFUNC = 'CREATE' ) THEN DO
         IF ( ( OBJECT  = 'TABLESPACE' ) & ( NUMPARTS > Ø ) &,
```

```
            ( CNT_PRIQTY > MAX_COUNT ) ) |,
          ( ( OBJECT  = 'TABLESPACE' ) & ( NUMPARTS = Ø ) &,
            ( CNT_PRIQTY > 1 ) ) |,
          ( ( OBJECT  = 'INDEX' )      & ( PARTFLAG = 'YES' ) &,
            ( CNT_PRIQTY > MAX_COUNT ) ) |,
          ( ( OBJECT  = 'INDEX' )      & ( PARTFLAG = 'NO'  ) &,
            ( CNT_PRIQTY > 1 ) ) THEN DO
          LM = 'DDL ERROR # 41: TOO MANY "PRIQTY" CLAUSES'
          SM = 'DDL ERROR # 41'
          CALL DDLERROR YVAL, XL1, 'PRIQTY'
          END                        /* END OF MULTIPLE CHECKS    */
       END                           /* END OF DDLFUNC='CREATE'   */
    ELSE DO                          /* DDLFUNC \= 'CREATE'       */
       IF ( CNT_PRIQTY > 1 ) THEN DO
          LM = 'DDL ERROR # 42: TOO MANY "PRIQTY" CLAUSES'
          SM = 'DDL ERROR # 42'
          CALL DDLERROR YVAL, XL1, 'PRIQTY'
          END                        /* END OF CNT_PRIQTY > 1     */
       END                           /* END OF DDLFUNC \='CREATE' */
    IF ( DATATYPE(N_TOKEN,'NUM') \= 1 ) THEN DO
       LM = 'DDL ERROR # 43: INVALID PRIQTY VALUE. ENTER NUMBER'
       SM = 'DDL ERROR # 43'
       CALL DDLERROR YVAL, XL1, N_TOKEN
       END                           /* END OF DATATYPE CHECK     */
    IF ( LAST_USING_TYPE = 'VCAT') THEN DO
       LM = 'DDL ERROR # 44: "PRIQTY" INVALID FOR "USING VCAT"'
       SM = 'DDL ERROR # 44'
       CALL DDLERROR YVAL, XL1, 'PRIQTY'
       END                           /* END OF DATATYPE CHECK     */
    END                               /* END TOKEN='PRIQTY'        */
 WHEN TOKEN = 'SECQTY' THEN DO
    /* CHECK CLAUSE NOT FOUND IN SAME PART DDL SECTION           */
    IF ( CURRENT_PARTNO > Ø ) THEN DO
       IF ( PRE_SECQTY = CURRENT_PARTNO ) THEN DO
          LM = 'DDL ERROR # 45: "SECQTY" CLAUSE FOUND TWICE'
          SM = 'DDL ERROR # 45'
          CALL DDLERROR YVAL, XL1, 'SECQTY'
          END
       PRE_SECQTY = CURRENT_PARTNO
       END
    LAST_GOOD_TOKEN = TCNT + 1       /* SET VAR USED FOR CHECKS   */
    CNT_SECQTY = CNT_SECQTY + 1
    /* ALLOW MULTIPLE SECQTYS FOR CLUSTER IX AND PART-TS         */
    IF ( DDLFUNC = 'CREATE' ) THEN DO
       IF ( ( OBJECT  = 'TABLESPACE' ) & ( NUMPARTS > Ø ) &,
            ( CNT_SECQTY > MAX_COUNT ) ) |,
          ( ( OBJECT  = 'TABLESPACE' ) & ( NUMPARTS = Ø ) &,
            ( CNT_SECQTY > 1 ) ) |,
          ( ( OBJECT  = 'INDEX' )      & ( PARTFLAG = 'YES' ) &,
            ( CNT_SECQTY > MAX_COUNT ) ) |,
```

```
         ( ( OBJECT  = 'INDEX' )      & ( PARTFLAG = 'NO' ) &,
           ( CNT_SECQTY > 1 ) ) THEN DO
         LM = 'DDL ERROR # 46: TOO MANY "SECQTY" CLAUSES'
         SM = 'DDL ERROR # 46'
         CALL DDLERROR YVAL, XL1, 'SECQTY'
         END                        /* END OF MULTIPLE CHECKS    */
      END                           /* END OF DDLFUNC='CREATE'   */
   ELSE DO                          /* ELSE, DDLFUNC \= 'CREATE' */
      IF ( CNT_SECQTY > 1 ) THEN DO
         LM = 'DDL ERROR # 47: TOO MANY "SECQTY" CLAUSES'
         SM = 'DDL ERROR # 47'
         CALL DDLERROR YVAL, XL1, 'SECQTY'
         END                        /* END OF CNT_SECQTY > 1     */
      END                           /* END OF DDLFUNC \= CREATE  */
   IF ( DATATYPE(N_TOKEN,'NUM') \= 1 ) THEN DO
      LM = 'DDL ERROR # 48: INVALID SECQTY VALUE. ENTER NUMBER'
      SM = 'DDL ERROR # 48'
      CALL DDLERROR YVAL, XL1, N_TOKEN
      END                           /* END OF DATATYPE CHECK     */
   IF ( LAST_USING_TYPE = 'VCAT') THEN DO
      LM = 'DDL ERROR # 49: "SECQTY" INVALID FOR "USING VCAT"'
      SM = 'DDL ERROR # 49'
      CALL DDLERROR YVAL, XL1, 'SECQTY'
      END                           /* END OF DATATYPE CHECK     */
   END                               /* END TOKEN='SECQTY'       */
WHEN TOKEN = 'ERASE' THEN DO
   /* CHECK CLAUSE NOT FOUND IN SAME PART DDL SECTION          */
   IF ( CURRENT_PARTNO > Ø ) THEN DO
      IF ( PRE_ERASE = CURRENT_PARTNO ) THEN DO
         LM = 'DDL ERROR # 5Ø: "ERASE" CLAUSE FOUND TWICE'
         SM = 'DDL ERROR # 5Ø'
         CALL DDLERROR YVAL, XL1, 'ERASE'
         END
      PRE_ERASE = CURRENT_PARTNO
      END
   LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
   CNT_ERASE = CNT_ERASE + 1
   /* ALLOW MULTIPLE ERASE FOR CLUSTER INDEX AND PART TSPACES  */
   IF ( DDLFUNC = 'CREATE' ) THEN DO
      IF ( ( OBJECT  = 'TABLESPACE' ) & ( NUMPARTS > Ø ) &,
           ( CNT_ERASE > MAX_COUNT ) ) |,
         ( ( OBJECT  = 'TABLESPACE' ) & ( NUMPARTS = Ø ) &,
           ( CNT_ERASE > 1 ) ) |,
         ( ( OBJECT  = 'INDEX' )      & ( PARTFLAG = 'YES' ) &,
           ( CNT_ERASE > MAX_COUNT ) ) |,
         ( ( OBJECT  = 'INDEX' )      & ( PARTFLAG = 'NO' ) &,
           ( CNT_ERASE > 1 ) ) THEN DO
         LM = 'DDL ERROR # 51: TOO MANY "ERASE" CLAUSES'
         SM = 'DDL ERROR # 51'
         CALL DDLERROR YVAL, XL1, 'ERASE'
```

```
         END                        /* END OF MULTIPLE CHECKS    */
      END                           /* END OF DDLFUNC='CREATE'   */
   ELSE DO                          /* ELSE, DDLFUNC \= 'CREATE' */
      IF ( CNT_ERASE > 1 ) THEN DO
         LM = 'DDL ERROR # 52: TOO MANY "ERASE" CLAUSES'
         SM = 'DDL ERROR # 52'
         CALL DDLERROR YVAL, XL1, 'ERASE'
         END                        /* END OF CNT_ERASE > 1      */
      END                           /* END OF DDLFUNC \= CREATE  */
   IF \( N_TOKEN = 'YES' | N_TOKEN = 'NO' ) THEN DO
      LM = 'ERASE VALUE MUST BE YES OR NO, NOT' N_TOKEN
      SM = 'WRONG ERASE VALUE'
      CALL DDLERROR YVAL, XL1, 'ERASE', N_TOKEN
      END
   END                               /* END TOKEN='ERASE'        */
WHEN TOKEN = 'GBPCACHE' THEN DO
   /* CHECK CLAUSE NOT FOUND IN SAME PART DDL SECTION          */
   IF ( CURRENT_PARTNO > Ø ) THEN DO
      IF ( PRE_GBPCACHE = CURRENT_PARTNO ) THEN DO
         LM = 'DDL ERROR # 53: "GBPCACHE" CLAUSE FOUND TWICE'
         SM = 'DDL ERROR # 53'
         CALL DDLERROR YVAL, XL1, 'GBPCACHE'
         END
      PRE_GBPCACHE = CURRENT_PARTNO
      END
   LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
   CNT_GBPCACHE = CNT_GBPCACHE + 1
   /* ALLOW MULTIPLE GBPCACHES FOR CLUSTER IX AND PART-TS      */
   IF ( DDLFUNC = 'CREATE' ) THEN DO
      IF ( ( OBJECT  = 'TABLESPACE' ) & ( NUMPARTS > Ø ) &,
           ( CNT_GBPCACHE > MAX_COUNT ) ) |,
         ( ( OBJECT  = 'TABLESPACE' ) & ( NUMPARTS = Ø ) &,
           ( CNT_GBPCACHE > 1 ) ) |,
         ( ( OBJECT  = 'INDEX' )      & ( PARTFLAG = 'YES' ) &,
           ( CNT_GBPCACHE > MAX_COUNT ) ) |,
         ( ( OBJECT  = 'INDEX' )      & ( PARTFLAG = 'NO' ) &,
           ( CNT_GBPCACHE > 1 ) ) THEN DO
         LM = 'DDL ERROR # 54: TOO MANY "GBPCACHE" CLAUSES'
         SM = 'DDL ERROR # 54'
         CALL DDLERROR YVAL, XL1, 'GBPCACHE'
         END                        /* END OF MULTIPLE CHECKS    */
      END                           /* END OF DDLFUNC='CREATE'   */
   ELSE DO                          /* ELSE, DDLFUNC \= 'CREATE' */
      IF ( CNT_GBPCACHE > 1 ) THEN DO
         LM = 'DDL ERROR # 55: TOO MANY "GBPCACHE" CLAUSES'
         SM = 'DDL ERROR # 55'
         CALL DDLERROR YVAL, XL1, 'GBPCACHE'
         END                        /* END OF CNT_GBPCACHE > 1   */
      END                           /* END OF DDLFUNC \= CREATE  */
   IF \ (( N_TOKEN = 'CHANGED' ) | ( N_TOKEN = 'ALL' ) ) THEN DO
```

```
      LM = 'DDL ERROR # 56: GBPCACHE MUST BE "CHANGED" OR "ALL"'
      SM = 'DDL ERROR # 56'
      CALL DDLERROR YVAL, XL1, 'GBPCACHE'
      END                          /* END OF KEYWORD CHECK      */
   END                              /* END TOKEN='GBPCACHE'      */
WHEN TOKEN = 'FREEPAGE' THEN DO
   /* CHECK CLAUSE NOT FOUND IN SAME PART DDL SECTION           */
   IF ( CURRENT_PARTNO > Ø ) THEN DO
      IF ( PRE_FREEPAGE = CURRENT_PARTNO ) THEN DO
         LM = 'DDL ERROR # 57: "FREEPAGE" CLAUSE FOUND TWICE'
         SM = 'DDL ERROR # 57'
         CALL DDLERROR YVAL, XL1, 'FREEPAGE'
         END
      PRE_FREEPAGE = CURRENT_PARTNO
      END
   LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
   CNT_FREEPAGE = CNT_FREEPAGE+1
   /* ALLOW MULTIPLE FREEPAGES FOR CLUSTER IX AND PART-TS       */
   IF ( DDLFUNC = 'CREATE' ) THEN DO
      IF ( ( OBJECT  = 'TABLESPACE' ) & ( NUMPARTS > Ø ) &,
           ( CNT_FREEPAGE > MAX_COUNT ) ) |,
         ( ( OBJECT  = 'TABLESPACE' ) & ( NUMPARTS = Ø ) &,
           ( CNT_FREEPAGE > 1 ) ) |,
         ( ( OBJECT  = 'INDEX' )      & ( PARTFLAG = 'YES' ) &,
           ( CNT_FREEPAGE > MAX_COUNT ) ) |,
         ( ( OBJECT  = 'INDEX' )      & ( PARTFLAG = 'NO' ) &,
           ( CNT_FREEPAGE > 1 ) ) THEN DO
         LM = 'DDL ERROR # 58: TOO MANY "FREEPAGE" CLAUSES'
         SM = 'DDL ERROR # 58'
         CALL DDLERROR YVAL, XL1, 'FREEPAGE'
         END                       /* END OF MULTIPLE CHECKS    */
      END                          /* END OF DDLFUNC='CREATE'   */
   ELSE DO                         /* ELSE, DDLFUNC \= 'CREATE' */
      IF ( CNT_FREEPAGE > 1 ) THEN DO
         LM = 'DDL ERROR # 59: TOO MANY "FREEPAGE" CLAUSES'
         SM = 'DDL ERROR # 59'
         CALL DDLERROR YVAL, XL1, 'FREEPAGE'
         END                       /* END OF CNT_FREEPAGE > 1   */
      END                          /* END OF DDLFUNC \= CREATE  */
   IF ( DATATYPE(N_TOKEN,'NUM') \= 1 ) THEN DO
      LM = 'DDL ERROR # 6Ø: INVALID FREEPAGE PARM. ENTER NUMBER'
      SM = 'DDL ERROR # 6Ø'
      CALL DDLERROR YVAL, XL1, 'FREEPAGE', N_TOKEN
      END                          /* END OF DATATYPE CHECK     */
   IF ( ( N_TOKEN < Ø ) | ( N_TOKEN > 255 ) ) THEN DO
      LM = 'DDL ERROR # 61: FREEPAGE MUST BE BETWEEN Ø & 255'
      SM = 'DDL ERROR # 61'
      CALL DDLERROR YVAL, XL1, 'FREEPAGE', N_TOKEN
      END                          /* END OF RANGE CHECK        */
   END                              /* END TOKEN='FREEPAGE'      */
```

```
WHEN TOKEN = 'PCTFREE' THEN DO
   /* CHECK CLAUSE NOT FOUND IN SAME PART DDL SECTION          */
   IF ( CURRENT_PARTNO > Ø ) THEN DO
      IF ( PRE_PCTFREE = CURRENT_PARTNO ) THEN DO
         LM = 'DDL ERROR # 62: "PCTFREE" CLAUSE FOUND TWICE'
         SM = 'DDL ERROR # 62'
         CALL DDLERROR YVAL, XL1, 'PCTFREE'
         END
      PRE_PCTFREE = CURRENT_PARTNO
      END
   LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
   CNT_PCTFREE = CNT_PCTFREE + 1
   /* ALLOW MULTIPLE PCTFREE   FOR CLUSTER IX AND PART-TS      */
   IF ( DDLFUNC = 'CREATE' ) THEN DO
      IF ( ( OBJECT  = 'TABLESPACE' ) & ( NUMPARTS > Ø ) &,
           ( CNT_PCTFREE > MAX_COUNT ) ) |,
         ( ( OBJECT  = 'TABLESPACE' ) & ( NUMPARTS = Ø ) &,
           ( CNT_PCTFREE > 1 ) ) |,
         ( ( OBJECT  = 'INDEX' )      & ( PARTFLAG = 'YES' ) &,
           ( CNT_PCTFREE > MAX_COUNT ) ) |,
         ( ( OBJECT  = 'INDEX' )      & ( PARTFLAG = 'NO' ) &,
           ( CNT_PCTFREE > 1 ) ) THEN DO
         LM = 'DDL ERROR # 63: TOO MANY "PCTFREE" CLAUSES'
         SM = 'DDL ERROR # 63'
         CALL DDLERROR YVAL, XL1, 'PCTFREE'
         END                      /* END OF MULTIPLE CHECKS    */
      END                         /* END OF DDLFUNC='CREATE'   */
   ELSE DO                        /* ELSE, DDLFUNC \= 'CREATE' */
      IF ( CNT_PCTFREE > 1 ) THEN DO
         LM = 'DDL ERROR # 64: TOO MANY "PCTFREE" CLAUSES'
         SM = 'DDL ERROR # 64'
         CALL DDLERROR YVAL, XL1, 'PCTFREE'
         END                      /* END OF CNT_PCTFREE > 1    */
      END                         /* END OF DDLFUNC \= CREATE  */
   IF ( DATATYPE(N_TOKEN,'NUM') \= 1 ) THEN DO
      LM = 'DDL ERROR # 65: INVALID PCTFREE VALUE. ENTER NUMBER'
      SM = 'DDL ERROR # 65'
      CALL DDLERROR YVAL, XL1, 'PCTFREE', N_TOKEN
      END                         /* END OF DATATYPE CHECK     */
   IF ( ( N_TOKEN < Ø ) | ( N_TOKEN > 99 ) ) THEN DO
      LM = 'DDL ERROR # 66: PCTFREE VALUE MUST BE BETWEEN Ø & 99'
      SM = 'DDL ERROR # 66'
      CALL DDLERROR YVAL, XL1, 'PCTFREE', N_TOKEN
      END                         /* END OF RANGE CHECK        */
   END                             /* END TOKEN='PCTFREE'      */
/*-------------------------------------------------------------*/
/*   COMPRESS CLAUSE CAN ONLY BE FOUND IN TABLESPACES.         */
/*   IT CAN ONLY BE FOUND ONCE IN ALTER TABLESPACE STMTS.      */
/*   IN CREATE PARTITIONED TABLESPACE STATEMENTS, IT CAN BE    */
/*   FOUND WITHIN THE PARTS SPECIFICATIONS AND/OR LATER.       */
```

```
/*-------------------------------------------------------------*/
WHEN TOKEN = 'COMPRESS' THEN DO
   /* CHECK CLAUSE NOT FOUND IN SAME PART DDL SECTION          */
   IF ( CURRENT_PARTNO > Ø ) THEN DO
      IF ( PRE_COMPRESS = CURRENT_PARTNO ) THEN DO
         LM = 'DDL ERROR # 67: "COMPRESS" CLAUSE FOUND TWICE'
         SM = 'DDL ERROR # 67'
         CALL DDLERROR YVAL, XL1, 'COMPRESS'
         END
      PRE_COMPRESS = CURRENT_PARTNO
      END
   LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
   CNT_COMPRESS = CNT_COMPRESS + 1
   IF ( OBJECT \= 'TABLESPACE' ) THEN DO
      LM = 'DDL ERROR # 68: "COMPRESS" ONLY VALID FOR TABLE-SP'
      SM = 'DDL ERROR # 68'
      CALL DDLERROR YVAL, XL1, 'COMPRESS'
      END                          /* END OF CNT_COMPRESS > 1  */
   IF ( DDLFUNC = 'CREATE' ) THEN DO
      IF ( PARTFLAG = 'YES' ) THEN DO
         IF ( MAX_COUNT = Ø ) THEN DO
            LM = 'DDL ERROR # 69: "COMPRESS" CLAUSE OUT OF PLACE'
            SM = 'DDL ERROR # 69'
            CALL DDLERROR YVAL, XL1, 'COMPRESS'
            END
         IF ( CNT_COMPRESS > MAX_COUNT ) THEN DO
            LM = 'DDL ERROR # 7Ø: TOO MANY "COMPRESS" CLAUSES'
            SM = 'DDL ERROR # 7Ø'
            CALL DDLERROR YVAL, XL1, 'COMPRESS'
            END                    /* END CNT_COMPRESS > MAX_.. */
         END
      ELSE DO                      /* ELSE OF PARTFLAG = 'YES'  */
         IF ( CNT_COMPRESS > 1 ) THEN DO
            LM = 'DDL ERROR # 71: TOO MANY "COMPRESS" CLAUSES'
            SM = 'DDL ERROR # 71'
            CALL DDLERROR YVAL, XL1, 'COMPRESS'
            END                    /* END CNT_COMPRESS > 1      */
         ELSE NOP                  /* NO PROBLEM, KEEP GOING    */
         END                       /* END OF ELSE PARTFLAG='YES'*/
      END                          /* END OF DDLFUNC= 'CREATE'  */
   ELSE DO                         /* ELSE, DDLFUNC \= 'CREATE' */
      IF ( CNT_COMPRESS > 1 ) THEN DO
         LM = 'DDL ERROR # 72: TOO MANY "COMPRESS" CLAUSES'
         SM = 'DDL ERROR # 72'
         CALL DDLERROR YVAL, XL1, 'COMPRESS'
         END                       /* END OF CNT_COMPRESS > 1  */
      ELSE NOP
      END                          /* END OF ELSE SECTION   1  */
   IF ( DDLFUNC = 'CREATE' ) & ( PARTFLAG = 'YES' ) THEN DO
      IF ( MAX_COUNT = Ø ) THEN DO
```

```
            LM = 'DDL ERROR # 73: "COMPRESS" CLAUSE OUT OF PLACE'
            SM = 'DDL ERROR # 73'
            CALL DDLERROR YVAL, XL1, 'COMPRESS'
            END
         END                          /* END OF DDLFUNC='CREATE'   */
      IF \( N_TOKEN = 'YES' | N_TOKEN = 'NO' ) THEN DO
         LM = 'DDL ERROR # 74: COMPRESS VALUE MUST BE "YES" OR "NO"'
         SM = 'DDL ERROR # 74'
         CALL DDLERROR YVAL, XL1, 'COMPRESS',N_TOKEN
         END                           /* END CHECK N_TOKEN VALUE  */
      END                              /* END TOKEN='COMPRESS'     */
   /*-------------------------------------------------------------*/
   /*   GROUP LOGIC FOR TOKENS THAT SHOULD BE FOUND ONLY ONCE.    */
   /*-------------------------------------------------------------*/
   WHEN TOKEN = 'SEGSIZE' THEN DO
      LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
      CNT_SEGSIZE = CNT_SEGSIZE + 1
      IF CNT_SEGSIZE > 1 THEN DO
         LM = 'DDL ERROR # 75: TOO MANY "SEGSIZE" CLAUSES'
         SM = 'DDL ERROR # 75'
         CALL DDLERROR YVAL, CL1, 'SEGSIZE', 'SEGSIZE'
         END                           /* END OF CNT_SEGSIZE > 1   */
      IF ( DATATYPE(N_TOKEN,'NUM') \= 1 ) THEN DO
         LM = 'DDL ERROR # 76: INVALID SEGSIZE PARM. MUST BE #'
         SM = 'DDL ERROR # 76'
         CALL DDLERROR YVAL, CL1, 'SEGSIZE', N_TOKEN
         END                          /* END DATATYPE CHECK        */
      END                              /* END TOKEN='SEGSIZE'       */
   WHEN TOKEN = 'IN'  THEN DO
      LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
      CNT_IN_DB = CNT_IN_DB + 1
      IF CNT_IN_DB > 1 THEN DO
         LM = 'DDL ERROR # 77: TOO MANY "IN" CLAUSES'
         SM = 'DDL ERROR # 77'
         CALL DDLERROR YVAL, XL1, 'IN ', 'IN '
         END                           /* END OF CNT_IN_DB   > 1   */
      CALL CHECK_VALID_NAME N_TOKEN, 'IN '   /* VALIDATE DB NAME  */
      END                              /* END TOKEN='IN'            */
   WHEN TOKEN = 'ON'  THEN DO
      CNT_ON_TABLE = CNT_ON_TABLE + 1
      IF CNT_ON_TABLE > 1 THEN DO
         LM = 'DDL ERROR # 78: TOO MANY "ON" CLAUSES'
         SM = 'DDL ERROR # 78'
         CALL DDLERROR YVAL, XL1, 'ON'
         END                           /* END OF CNT_ON_TABLE > 1  */
      NEXT_RPAREN = WORDPOS(')',DDLSTR,TCNT)  /* GET POS OF ')'   */
      LAST_GOOD_TOKEN = NEXT_RPAREN   /* SET VAR USED FOR CHECKS  */
      END                              /* END TOKEN='ON'            */
   WHEN TOKEN = 'ROSHARE' THEN DO
      LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
```

```
      CNT_ROSHARE = CNT_ROSHARE + 1
      IF CNT_ROSHARE > 1 THEN DO
         LM = 'DDL ERROR # 79: TOO MANY "ROSHARE" CLAUSES'
         SM = 'DDL ERROR # 79'
         CALL DDLERROR YVAL, CL1, 'ROSHARE', 'ROSHARE'
         END                          /* END OF CNT_ROSHARE > 1   */
      IF \( N_TOKEN = 'OWNER' | N_TOKEN = 'NONE' ) THEN DO
         LM = 'ROSHARE VALUE MUST BE OWNER OR NONE, NOT ' N_TOKEN
         SM = 'WRONG ROSHARE VALUE'
         CALL DDLERROR YVAL, CL1, 'ROSHARE', N_TOKEN, CNT_ROSHARE
         END
      END                             /* END TOKEN='ROSHARE'      */
   WHEN TOKEN = 'WORKFILE' & PREV_TOKEN = 'AS' THEN DO
      CNT_WORKFILE = CNT_WORKFILE + 1
      LAST_GOOD_TOKEN = TCNT          /* SET VAR USED FOR CHECKS  */
      IF CNT_WORKFILE > 1 THEN DO
         LM = 'DDL ERROR # 8Ø: TOO MANY "WORKFILE" CLAUSES'
         SM = 'DDL ERROR # 8Ø'
         CALL DDLERROR YVAL, CL1, 'WORKFILE', 'WORKFILE'
         END                          /* END OF CNT_WORKFILE > 1  */
      IF ( N_TOKEN = 'FOR' ) THEN DO  /* CHECK MEMBER NAME        */
         CALL CHECK_VALID_NAME N2_TOKEN, 'WORKFILE'
         LAST_GOOD_TOKEN = TCNT + 2   /* SET VAR USED FOR CHECKS  */
         END                          /* END N_TOKEN = 'WORKFILE' */
      END                             /* END TOKEN='WORKFILE'     */
   WHEN TOKEN = 'AS' THEN DO
      IF ( N_TOKEN = 'WORKFILE' ) THEN DO
         LAST_GOOD_TOKEN = TCNT       /* SET VAR USED FOR CHECKS  */
         ITERATE
         END                          /* END N_TOKEN = 'WORKFILE' */
      ELSE DO
         LM = 'DDL ERROR # 81: AS MUST BE FOLLOWED BY WORKFILE'
         SM = 'DDL ERROR # 81'
         CALL DDLERROR YVAL, CL1, 'AS', N_TOKEN
         END                          /* END ELSE STATEMENT       */
      END                             /* END TOKEN='AS'           */
   WHEN TOKEN = 'STOGROUP' THEN DO
      IF ( OBJECT = 'DATABASE' ) THEN DO
         LAST_GOOD_TOKEN = TCNT + 1   /* SET VAR USED FOR CHECKS  */
         CNT_STOGROUP = CNT_STOGROUP + 1
         IF ( CNT_STOGROUP > 1 ) THEN DO
            LM = 'DDL ERROR # 82: TOO MANY "STOGROUP" CLAUSES'
            SM = 'DDL ERROR # 82'
            CALL DDLERROR YVAL, CL1, PREV_TOKEN, 'STOGROUP'
            END                       /* END ELSE STATEMENT       */
         END                          /* END OBJECT='DATABASE'    */
      END                             /* END TOKEN='STOGROUP'     */
   WHEN TOKEN = 'LARGE' THEN DO
      LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
      CNT_LARGE = CNT_LARGE + 1
```

```
   IF CNT_LARGE > 1 THEN DO
      LM = 'DDL ERROR # 83: TOO MANY "LARGE" CLAUSES'
      SM = 'DDL ERROR # 83'
      CALL DDLERROR YVAL, CL1, 'LARGE', 'LARGE'
      END                          /* END OF CNT_LARGE > 1     */
   END                             /* END TOKEN='LARGE'        */
WHEN TOKEN = 'BUFFERPOOL' THEN DO
   LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
   CNT_BUFFERPOOL = CNT_BUFFERPOOL + 1
   IF CNT_BUFFERPOOL > 1 THEN DO
      LM = 'DDL ERROR # 84: TOO MANY "BUFFERPOOL" CLAUSES'
      SM = 'DDL ERROR # 84'
      CALL DDLERROR YVAL, CL1, 'BUFFERPOOL', 'BUFFERPOOL'
      END                          /* END OF CNT_BUFFERPOOL > 1*/
   CALL CHECK_BUFFERPOOL_PARM N_TOKEN    /* VALIDATE DB_BPNAME */
   END                             /* END TOKEN='BUFFERPOOL'   */
WHEN TOKEN = 'LOCKSIZE' THEN DO
   LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
   CNT_LOCKSIZE = CNT_LOCKSIZE + 1
   IF CNT_LOCKSIZE > 1 THEN DO
      LM = 'DDL ERROR # 85: TOO MANY "LOCKSIZE" CLAUSES'
      SM = 'DDL ERROR # 85'
      CALL DDLERROR YVAL, CL1, 'LOCKSIZE', '',CNT_LOCKSIZE
      END                          /* END OF CNT_LOCKSIZE > 1  */
   IF \( N_TOKEN= 'ANY' | N_TOKEN = 'TABLESPACE' |,
         N_TOKEN= 'TABLE' | N_TOKEN = 'PAGE' |,
         N_TOKEN = 'ROW' ) THEN DO
      LM = 'DDL ERROR # 86: INVALID LOCKSIZE PARM. CHECK PARM'
      SM = 'DDL ERROR # 86'
      CALL DDLERROR YVAL, CL1, 'LOCKSIZE', N_TOKEN
      END                          /* END OF CHECK LOCKSIZE KWD*/
   LOCKSIZE_PARM = N_TOKEN         /* SAVE PARM FOR LATER PROC */
   END                             /* END TOKEN='LOCKSIZE'     */
WHEN TOKEN = 'LOCKMAX' THEN DO
   LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
   CNT_LOCKMAX = CNT_LOCKMAX + 1
   IF CNT_LOCKMAX > 1 THEN DO
      LM = 'DDL ERROR # 87: TOO MANY "LOCKMAX" CLAUSES'
      SM = 'DDL ERROR # 87'
      CALL DDLERROR YVAL, CL1, 'LOCKMAX', '',CNT_LOCKMAX
      END                          /* END OF CNT_LOCKMAX > 1   */
   IF ( N_TOKEN \= 'SYSTEM') & ,
      ( (N_TOKEN\='SYSTEM') &,
        (DATATYPE(N_TOKEN,'NUM') \= 1 ) ) THEN DO
      LM = 'DDL ERROR # 88: INVALID LOCKMAX PARM. SYSTEM OR #'
      SM = 'DDL ERROR # 88'
      CALL DDLERROR YVAL, CL1, 'LOCKMAX', N_TOKEN
      END                          /* END OF CHECK LOCKMAX KWRD*/
   END                             /* END TOKEN='LOCKMAX'      */
WHEN TOKEN = 'LOCKPART' THEN DO
```

```
      LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
      CNT_LOCKPART = CNT_LOCKPART + 1
      IF CNT_LOCKPART > 1 THEN DO
         LM = 'DDL ERROR # 89: TOO MANY "LOCKPART" CLAUSES'
         SM = 'DDL ERROR # 89'
         CALL DDLERROR YVAL, CL1, 'LOCKPART', 'LOCKPART'
         END                         /* END OF CNT_LOCKPART > 1   */
      IF \( N_TOKEN = 'YES' | N_TOKEN = 'NO' ) THEN DO
         LM = 'LOCKPART VALUE MUST BE YES OR NO, NOT' N_TOKEN
         SM = 'WRONG LOCKPART VALUE'
         CALL DDLERROR YVAL,CL1,'LOCKPART',N_TOKEN,CNT_LOCKPART
         END
      LOCKPART_PARM = N_TOKEN         /* SAVE PARM FOR LATER PROC */
      END                             /* END TOKEN='LOCKPART'     */
   WHEN TOKEN = 'CLOSE' THEN DO
      LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
      CNT_CLOSE = CNT_CLOSE + 1
      IF CNT_CLOSE > 1 THEN DO
         LM = 'DDL ERROR # 90: TOO MANY "CLOSE" CLAUSES'
         SM = 'DDL ERROR # 90'
         CALL DDLERROR YVAL, CL1, 'CLOSE', 'CLOSE'
         END                          /* END OF CNT_CLOSE > 1     */
      IF \( N_TOKEN = 'YES' | N_TOKEN = 'NO' ) THEN DO
         LM = 'CLOSE VALUE MUST BE YES OR NO, NOT' N_TOKEN
         SM = 'WRONG CLOSE VALUE'
         CALL DDLERROR YVAL, CL1, 'CLOSE', N_TOKEN, CNT_CLOSE
         END
      END                             /* END TOKEN='CLOSE'        */
   WHEN TOKEN = 'DSETPASS' THEN DO
      LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
      CNT_DSETPASS = CNT_DSETPASS + 1
      IF CNT_DSETPASS > 1 THEN DO
         LM = 'DDL ERROR # 91: TOO MANY "DSETPASS" CLAUSES'
         SM = 'DDL ERROR # 91'
         CALL DDLERROR YVAL, CL1, 'DSETPASS', 'DSETPASS'
         END                          /* END OF CNT_DSETPASS > 1  */
      CALL CHECK_VALID_NAME N_TOKEN, 'DSETPASS' /* VALIDATE NAME  */
      END                             /* END TOKEN='DSETPASS'     */
   WHEN TOKEN = 'CCSID' THEN DO
      LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
      CNT_CCSID = CNT_CCSID + 1
      IF CNT_CCSID > 1 THEN DO
         LM = 'DDL ERROR # 92: TOO MANY "CCSID" CLAUSES'
         SM = 'DDL ERROR # 92'
         CALL DDLERROR YVAL, CL1, 'CCSID', 'CCSID'
         END                          /* END OF CNT_CCSID > 1     */
      IF \( N_TOKEN = 'ASCII' | N_TOKEN = 'EBCDIC' ) THEN DO
         LM = 'CCSID VALUE MUST BE ASCII OR EBCDIC, NOT' N_TOKEN
         SM = 'WRONG CCSID VALUE'
         CALL DDLERROR YVAL, CL1, 'CCSID', N_TOKEN, CNT_CCSID
```

```
      END
   END                              /* END TOKEN='CCSID'        */
WHEN TOKEN = 'MAXROWS' THEN DO
   LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
   CNT_MAXROWS = CNT_MAXROWS + 1
   IF CNT_MAXROWS > 1 THEN DO
      LM = 'DDL ERROR # 93: TOO MANY "MAXROWS" CLAUSES'
      SM = 'DDL ERROR # 93'
      CALL DDLERROR YVAL, CL1, 'MAXROWS', 'MAXROWS'
      END                           /* END OF CNT_MAXROWS > 1    */
   IF ( DATATYPE(N_TOKEN,'NUM') \= 1 ) THEN DO
      LM = 'DDL ERROR # 94: INVALID MAXROWS VALUE. ENTER NUMBER'
      SM = 'DDL ERROR # 94'
      CALL DDLERROR YVAL, CL1, 'MAXROWS', N_TOKEN
      END                           /* END OF DATATYPE CHECK     */
   IF ( ( N_TOKEN < 1 ) | ( N_TOKEN > 255 ) ) THEN DO
      LM = 'DDL ERROR # 95: MAXROWS VALUE RANGE IS 1 TO 255'
      SM = 'DDL ERROR # 95'
      CALL DDLERROR YVAL, CL1, 'MAXROWS', N_TOKEN
      END                           /* END OF RANGE CHECK        */
   END                              /* END TOKEN='MAXROWS'       */
WHEN TOKEN = 'CONVERT' THEN DO
   CNT_CONVERT = CNT_CONVERT + 1
   IF CNT_CONVERT > 1 THEN DO
      LM = 'DDL ERROR # 96: TOO MANY "CONVERT" CLAUSES'
      SM = 'DDL ERROR # 96'
      CALL DDLERROR YVAL, CL1, 'CONVERT', 'CONVERT'
      END                           /* END OF CNT_CONVERT > 1    */
   IF ( N_TOKEN \= 'TO' ) THEN DO
      LM = 'DDL ERROR # 97: "TO" MUST FOLLOW A "CONVERT"'
      SM = 'DDL ERROR # 97'
      CALL DDLERROR YVAL, CL1, 'CONVERT', N_TOKEN
      END                           /* END N_TOKEN \= 'TO'       */
   IF ( N2_TOKEN \= 'TYPE' ) THEN DO
      LM = 'DDL ERROR # 98: "TYPE" MUST FOLLOW A "TO"'
      SM = 'DDL ERROR # 98'
      CALL DDLERROR YVAL, CL1, 'TO', N_TOKEN
      END                           /* END N2_TOKEN \= 'TYPE'    */
   LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
   END                              /* END TOKEN='TO'            */
WHEN TOKEN = 'TYPE' THEN DO
   LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
   CNT_TYPE = CNT_TYPE + 1
   IF CNT_TYPE > 1 THEN DO
      LM = 'DDL ERROR # 99: TOO MANY "TYPE" CLAUSES'
      SM = 'DDL ERROR # 99'
      CALL DDLERROR YVAL, CL1, 'TYPE', 'TYPE'
      END                           /* END OF CNT_TYPE > 1       */
   IF ( DATATYPE(N_TOKEN,'NUM') \= 1 ) THEN DO
      LM = 'DDL ERROR # 100: INDEX TYPE MUST BE NUMERIC'
```

```
      SM = 'DDL ERROR # 100'
      CALL DDLERROR YVAL, CL1, 'TYPE', N_TOKEN
      END                          /* END OF DATATYPE CHECK     */
   IF \ ( ( N_TOKEN = 1 ) | ( N_TOKEN = 2 ) ) THEN DO
      LM = 'DDL ERROR # 101: INDEX TYPE MUST BE 1 OR 2 '
      SM = 'DDL ERROR # 101'
      CALL DDLERROR YVAL, CL1, 'TYPE', N_TOKEN
      END                          /* END OF VALUES CHECK       */
   END                              /* END TOKEN='TYPE'          */
WHEN TOKEN = 'UNIQUE' THEN DO
   LAST_GOOD_TOKEN = TCNT + 0     /* SET VAR USED FOR CHECKS   */
   CNT_UNIQUE = CNT_UNIQUE + 1
   IF CNT_UNIQUE > 1 THEN DO
      LM = 'DDL ERROR # 102: TOO MANY "UNIQUE" CLAUSES'
      SM = 'DDL ERROR # 102'
      CALL DDLERROR YVAL, CL1, 'UNIQUE', 'UNIQUE'
      END                          /* END OF CNT_UNIQUE > 1     */
   END                              /* END TOKEN='UNIQUE'        */
WHEN TOKEN = 'WHERE' THEN DO
   IF ( N_TOKEN = 'NOT' ) & ( N2_TOKEN = 'NULL' ) &,
      ( PREV_TOKEN = 'UNIQUE' ) THEN DO
      LAST_GOOD_TOKEN = TCNT + 2   /* SET VAR USED FOR CHECKS  */
      ITERATE
      END                          /* END N_TOKEN = 'WORKFILE' */
   ELSE DO
      LM = 'DDL ERROR # 103: SYNTAX IS --> UNIQUE WHERE NOT NULL'
      SM = 'DDL ERROR # 103'
      CALL DDLERROR YVAL, CL1, 'WHERE', N_TOKEN
      END                          /* END ELSE STATEMENT        */
   END                              /* END TOKEN='WHERE'         */
WHEN TOKEN = 'CLUSTER' THEN DO
   LAST_GOOD_TOKEN = TCNT + 0     /* SET VAR USED FOR CHECKS   */
   CNT_CLUSTER = CNT_CLUSTER + 1
   IF CNT_CLUSTER > 1 THEN DO
      LM = 'DDL ERROR # 104: TOO MANY "CLUSTER" CLAUSES'
      SM = 'DDL ERROR # 104'
      CALL DDLERROR YVAL, CL1, 'CLUSTER', 'CLUSTER'
      END                          /* END OF CNT_CLUSTER > 1    */
   END                              /* END TOKEN='CLUSTER'       */
WHEN TOKEN = 'SUBPAGES' THEN DO
   LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
   CNT_SUBPAGES = CNT_SUBPAGES + 1
   IF CNT_SUBPAGES > 1 THEN DO
      LM = 'DDL ERROR # 105: TOO MANY "SUBPAGES" CLAUSES'
      SM = 'DDL ERROR # 105'
      CALL DDLERROR YVAL, CL1, 'SUBPAGES', 'SUBPAGES'
      END                          /* END OF CNT_SUBPAGES > 1   */
   IF ( DATATYPE(N_TOKEN,'NUM') \= 1) THEN DO
      LM = 'DDL ERROR # 106: INVALID SUB-PAGE SIZE.  MUST BE #'
      SM = 'DDL ERROR # 106'
```

```
      CALL DDLERROR YVAL, CL1, 'SUBPAGES', N_TOKEN
      END                          /* END CHECK SUBPAGES VALUE  */
   IF \ ( (N_TOKEN =  1) | (N_TOKEN =  2) | (N_TOKEN =  4) | ,
        (N_TOKEN =  8) | (N_TOKEN = 16) ) THEN DO
      LM = 'DDL ERROR # 107: SUB-PAGE SIZE MUST BE 1,2,4,8 OR 16'
      SM = 'DDL ERROR # 107'
      CALL DDLERROR YVAL, CL1, 'SUBPAGES', N_TOKEN
      END                          /* END CHECK SUBPAGES VALUE  */
   END                              /* END TOKEN='SUBPAGES'     */
WHEN TOKEN = 'DEFER' THEN DO
   LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
   CNT_DEFER = CNT_DEFER + 1
   IF CNT_DEFER > 1 THEN DO
      LM = 'DDL ERROR # 108: TOO MANY "DEFER" CLAUSES'
      SM = 'DDL ERROR # 108'
      CALL DDLERROR YVAL, CL1, 'DEFER', 'DEFER'
      END                          /* END OF CNT_DEFER > 1      */
   IF \( N_TOKEN = 'YES' | N_TOKEN = 'NO' ) THEN DO
      LM = 'DEFER VALUE MUST BE YES OR NO, NOT' N_TOKEN
      SM = 'WRONG DEFER VALUE'
      CALL DDLERROR YVAL, CL1, 'DEFER', N_TOKEN, CNT_CLOSE
      END
   END                              /* END TOKEN='DEFER'         */
WHEN TOKEN = 'PIECESIZE' THEN DO
   LAST_GOOD_TOKEN = TCNT + 1     /* SET VAR USED FOR CHECKS   */
   CNT_PIECESIZE = CNT_PIECESIZE + 1
   IF CNT_PIECESIZE > 1 THEN DO
      LM = 'DDL ERROR # 109: TOO MANY "PIECESIZE" CLAUSES'
      SM = 'DDL ERROR # 109'
      CALL DDLERROR YVAL, CL1, 'PIECESIZE', 'PIECESIZE'
      END                          /* END OF CNT_PIECESIZE > 1  */
   IF ( DATATYPE(N_TOKEN,'NUM') \= 1) THEN DO
      TSFX = RIGHT(N_TOKEN,1)
      IF (TSFX = 'K' ) | ( TSFX = 'M' ) | ( TSFX = 'G' ) THEN DO
         TLEN = LENGTH(N_TOKEN)
         N_TOKEN  = LEFT(N_TOKEN,(TLEN-1))
         N2_TOKEN = TSFX
         END
      ELSE DO
         LM = 'DDL ERROR # 110: CAN NOT PARSE PIECESIZE INFO'
         SM = 'DDL ERROR # 110'
         CALL DDLERROR YVAL, CL1, 'PIECESIZE', N_TOKEN
         END
      END                          /* END OF DATATYPE CHECK     */
   IF ( DATATYPE(N_TOKEN,'NUM') \= 1) THEN DO
      LM = 'DDL ERROR # 111: PIECESIZE VALUE MUST BE NUMERIC'
      SM = 'DDL ERROR # 111'
      CALL DDLERROR YVAL, CL1, 'PIECESIZE', N_TOKEN
      END                          /* END OF DATATYPE CHECK     */
   IF \ ( N2_TOKEN='K' | N2_TOKEN='M' | N2_TOKEN='G' ) THEN DO
```

```
   LM = 'DDL ERROR # 112: PIECESIZE MUST BE FOLLOWED BY',
        'A "K","M", OR "G"'
   SM = 'DDL ERROR # 112'
   CALL DDLERROR YVAL, CL1, N_TOKEN, N2_TOKEN
   END                          /* END OF DATATYPE CHECK     */
IF ( N2_TOKEN = 'K') THEN DO
   IF ( ( N_TOKEN < 256 ) | ( N_TOKEN > 4194304 ) ) THEN DO
      LM = 'DDL ERROR # 113: PIECESIZE MUST BE BETWEEN 256',
           ' & 4194304'
      SM = 'DDL ERROR # 113'
      CALL DDLERROR YVAL, CL1, 'PIECESIZE', N_TOKEN
      END                       /* END OF N_TOKEN CHECK      */
   WORKSIZE = N_TOKEN
   DO WHILE ( ( WORKSIZE >= 2) & ( ( WORKSIZE // 2 ) = 0 ) )
      WORKSIZE = WORKSIZE / 2
      END
   IF ( WORKSIZE \= 1) THEN DO
      LM = 'DDL ERROR # 114: PIECESIZE MUST BE A POWER OF 2'
      SM = 'DDL ERROR # 114'
      CALL DDLERROR YVAL, CL1, 'PIECESIZE', N_TOKEN
      END                       /* END OF WORKSIZE \= 0      */
   END                          /* END OF N2_TOKEN = 'K'     */
IF ( N2_TOKEN = 'M') THEN DO
   IF ( ( N_TOKEN < 1 ) | ( N_TOKEN > 4096 ) ) THEN DO
      LM = 'DDL ERROR # 115: PIECESIZE MUST BE BETWEEN 1',
           ' & 4096'
      SM = 'DDL ERROR # 115'
      CALL DDLERROR YVAL, CL1, 'PIECESIZE', N_TOKEN
      END                       /* END OF WORKSIZE \= 0      */
   WORKSIZE = N_TOKEN
   DO WHILE ( ( WORKSIZE >= 2) & ( ( WORKSIZE // 2 ) = 0 ) )
      WORKSIZE = WORKSIZE / 2
      END
   IF ( WORKSIZE \= 1) THEN DO
      LM = 'DDL ERROR # 116: PIECESIZE MUST BE A POWER OF 2'
      SM = 'DDL ERROR # 116'
      CALL DDLERROR YVAL, CL1, 'PIECESIZE', N_TOKEN
      END                       /* END OF WORKSIZE \= 0      */
   END                          /* END OF N2_TOKEN = 'K'     */
IF ( N2_TOKEN = 'G') THEN DO
   IF ( ( N_TOKEN < 1 ) | ( N_TOKEN > 4 ) ) THEN DO
      LM = 'DDL ERROR # 117: PIECESIZE MUST BE BETWEEN 1 & 4'
      SM = 'DDL ERROR # 117'
      CALL DDLERROR YVAL, CL1, 'PIECESIZE', N_TOKEN
      END                       /* END OF WORKSIZE \= 0      */
   WORKSIZE = N_TOKEN
   DO WHILE ( ( WORKSIZE >= 2) & ( ( WORKSIZE // 2 ) = 0 ) )
      WORKSIZE = WORKSIZE / 2
      END
   IF ( WORKSIZE \= 1) THEN DO
```

```
               LM = 'DDL ERROR # 118: PIECESIZE MUST BE A POWER OF 2'
               SM = 'DDL ERROR # 118'
               CALL DDLERROR YVAL, CL1, 'PIECESIZE', N_TOKEN
               END                        /* END OF WORKSIZE \= Ø      */
            END                           /* END OF N2_TOKEN = 'K'     */
         END                               /* END TOKEN='PIECESIZE'     */
      /*-------------------------------------------------------------*/
      /* NO OTHER SPECIFIC CLAUSES WERE FOUND. CHECK IF THE VARIABLE */
      /* LAST_GOOD_TOKEN IS GREATER THAN THE CURRENT TCNT VALUE. IF  */
      /* GREATER, TCNT IS STILL POINTING AT THE PARAMETERS OF A VALID*/
      /* CLAUSE.  IF IT IS LESS, CHECK IF THE OBJECT IS AN INDEX, OR */
      /* IF THE TOKEN IS A VALID LIST OF TOKENS. IF NOT, ISSUE MSG   */
      /*-------------------------------------------------------------*/
      OTHERWISE DO                         /* OTHERWISE CLAUSE         */
         IF ( ( TOKEN = 'CREATE'     ) | ,
              ( TOKEN = 'ALTER'      ) | ,
              ( TOKEN = 'LARGE'      ) ) THEN DO
            LAST_GOOD_TOKEN = TCNT
            ITERATE
            END
         IF ( ( TOKEN = 'DATABASE'   ) | ,
              ( TOKEN = 'TABLESPACE' ) | ,
              ( TOKEN = 'INDEX'      ) ) THEN DO
            LAST_GOOD_TOKEN = TCNT + 1
            ITERATE
            END
         IF ( (TCNT > LAST_GOOD_TOKEN) & (TCNT <= WORDCNT) ) THEN DO
            NOTEOUT = TOKEN '<-- POSSIBLE INCORRECT CLAUSE.'
            CALL WRITE_NOTE NOTEOUT, XL1        /* WRITE ERROR NOTE  */
            END                          /* END OF TCNT > L_G_TOKEN   */
         END                             /* END TOKEN OTHERWISE       */
      END                                /* END OF SELECT             */
   END                                   /* END OF DO FOR LOOP        */

/*---------------------------------------------------------------------*/
/*  PARAGRAPH : REVIEW THAT CREATE DATABASE USES CORRECT KEYWORDS.     */
/*---------------------------------------------------------------------*/
IF DDLFUNC = 'CREATE' &  OBJECT = 'DATABASE' THEN DO
   /* CHECK THAT THESE CLAUSES ARE NOT PRESENT IN THE DDL STATEMENT  */
   CLAUSE_NAME = ''
   IF ( CNT_CLOSE     > Ø ) THEN CLAUSE_NAME = 'CLOSE'
   IF ( CNT_CLUSTER   > Ø ) THEN CLAUSE_NAME = 'CLUSTER'
   IF ( CNT_CONVERT   > Ø ) THEN CLAUSE_NAME = 'CONVERT'
   IF ( CNT_DEFER     > Ø ) THEN CLAUSE_NAME = 'DEFER'
   IF ( CNT_DSETPASS  > Ø ) THEN CLAUSE_NAME = 'DSETPASS'
   IF ( CNT_FREEPAGE  > Ø ) THEN CLAUSE_NAME = 'FREEPAGE'
   IF ( CNT_GBPCACHE  > Ø ) THEN CLAUSE_NAME = 'GBPCACHE'
   IF ( CNT_IN_DB     > Ø ) THEN CLAUSE_NAME = 'IN'
   IF ( CNT_LARGE     > Ø ) THEN CLAUSE_NAME = 'LARGE'
   IF ( CNT_LOCKMAX   > Ø ) THEN CLAUSE_NAME = 'LOCKMAX'
```

```
   IF ( CNT_LOCKPART  > Ø ) THEN CLAUSE_NAME = 'LOCKPART'
   IF ( CNT_LOCKSIZE  > Ø ) THEN CLAUSE_NAME = 'LOCKSIZE'
   IF ( CNT_MAXROWS   > Ø ) THEN CLAUSE_NAME = 'MAXROWS'
   IF ( CNT_NUMPARTS  > Ø ) THEN CLAUSE_NAME = 'NUMPARTS'
   IF ( CNT_ON_TABLE  > Ø ) THEN CLAUSE_NAME = 'ON'
   IF ( CNT_PART      > Ø ) THEN CLAUSE_NAME = 'PART'
   IF ( CNT_PCTFREE   > Ø ) THEN CLAUSE_NAME = 'PCTFREE'
   IF ( CNT_PIECESIZE > Ø ) THEN CLAUSE_NAME = 'PIECESIZE'
   IF ( CNT_PRIQTY    > Ø ) THEN CLAUSE_NAME = 'PRIQTY'
   IF ( CNT_SECQTY    > Ø ) THEN CLAUSE_NAME = 'SECQTY'
   IF ( CNT_SEGSIZE   > Ø ) THEN CLAUSE_NAME = 'SEGSIZE'
   IF ( CNT_SUBPAGES  > Ø ) THEN CLAUSE_NAME = 'SUBPAGES'
   IF ( CNT_TYPE      > Ø ) THEN CLAUSE_NAME = 'TYPE'
   IF ( CNT_UNIQUE    > Ø ) THEN CLAUSE_NAME = 'UNIQUE'
   IF ( CLAUSE_NAME \= '' ) THEN DO
      SM = 'INVALID ' || CLAUSE_NAME
      LM = 'INVALID ' || CLAUSE_NAME || ,
           ' CLAUSE SPECIFIED IN CREATE DATABASE STATEMENT'
      CALL DDLERROR YVAL, CL1, OBJECT, CLAUSE_NAME
      END
   /*    CHECK FOR CLAUSES THAT ARE ONLY AVAILABLE FOR DB2 V5        */
   IF ( DB2VER = 'V4' ) & ( CNT_CCSID > Ø ) THEN DO
      LM = 'V5 CLAUSES ON DB2 V4 DDL.  MODIFY DDL OR PANEL VALUE'
      SM = 'CAN NOT USE CCSID'
      CALL DDLERROR YVAL, CL1, DDLFUNC, 'CCSID'
      END
   END               /* END FOR 'CREATE DATABASE' CLAUSE CHECKS      */

/*-------------------------------------------------------------------*/
/*  PARAGRAPH : REVIEW THAT CREATE TABLESPACE USES CORRECT KEYWORDS  */
/*-------------------------------------------------------------------*/
IF ( DDLFUNC = 'CREATE' ) & ( OBJECT = 'TABLESPACE' ) THEN DO
   CLAUSE_NAME = ''
   IF ( CNT_ON_TABLE  > Ø ) THEN CLAUSE_NAME = 'ON'
   IF ( CNT_ROSHARE   > Ø ) THEN CLAUSE_NAME = 'ROSHARE'
   IF ( CNT_PIECESIZE > Ø ) THEN CLAUSE_NAME = 'PIECESIZE'
   IF ( CNT_TYPE      > Ø ) THEN CLAUSE_NAME = 'TYPE'
   IF ( CNT_CLUSTER   > Ø ) THEN CLAUSE_NAME = 'CLUSTER'
   IF ( CNT_WORKFILE  > Ø ) THEN CLAUSE_NAME = 'WORKFILE'
   IF ( CNT_UNIQUE    > Ø ) THEN CLAUSE_NAME = 'UNIQUE'
   IF ( CNT_SUBPAGES  > Ø ) THEN CLAUSE_NAME = 'SUBPAGES'
   IF ( CNT_DEFER     > Ø ) THEN CLAUSE_NAME = 'DEFER'
   IF ( CLAUSE_NAME \= '' ) THEN DO
      SM = 'INVALID ' || CLAUSE_NAME
      LM = 'INVALID ' || CLAUSE_NAME || ,
           ' CLAUSE SPECIFIED IN CREATE TABLESPACE STATEMENT'
      CALL DDLERROR YVAL, CL1, OBJECT, CLAUSE_NAME
      END
   IF ( DB2VER = 'V4' ) THEN DO
      IF ( CNT_LARGE    > Ø ) THEN CLAUSE_NAME = 'LARGE'
```

```
      IF ( CNT_CCSID    > Ø ) THEN CLAUSE_NAME = 'CCSID'
      IF ( CNT_LOCKPART > Ø ) THEN CLAUSE_NAME = 'LOCKPART'
      IF ( CNT_MAXROWS  > Ø ) THEN CLAUSE_NAME = 'MAXROWS'
      IF ( CLAUSE_NAME \= '' ) THEN DO
         LM = 'V5 CLAUSE ON DB2 V4 DDL.  MODIFY DDL OR PANEL VALUE'
         SM = 'V5 CLAUSE>'||CLAUSE_NAME
         CALL DDLERROR YVAL, CL1, DDLFUNC, CLAUSE_NAME
         END
      END                          /* END IF DB2VER = 'V4' STMT       */
   /* CHECK COMPATABILITY OF LOCKPART AND LOCKSIZE PARAMETERS (V5)   */
   IF ( ( CNT_LOCKPART > Ø ) & ( CNT_LOCKSIZE > Ø ) ) THEN DO
      IF ( ( LOCKPART_PARM = 'YES' ) & ,
           ( LOCKSIZE_PARM = 'TABLESPACE') ) THEN DO
         LM = '"LOCKPART YES" AND "LOCKSIZE TABLESPACE" NOT COMPATIBLE'
         SM = 'INCOMPATIBLE PARMS'
         CALL DDLERROR YVAL, CL1, DDLFUNC, 'LOCKPART'
         END
      END                          /* END IF OF COMPATABILITY CHECK   */
   /* CHECK COMPATABILITY OF LARGE AND PARTITIONED TABLESPACE (V5)   */
   IF ( ( CNT_LARGE > Ø ) & ( CNT_NUMPARTS \= 1 ) ) THEN DO
      LM = '"LARGE" CLAUSE VALID ONLY ON PARTITIONED TABLESPACES'
      SM = 'INVALID CLAUSE'
      CALL DDLERROR YVAL, CL1, DDLFUNC, 'LARGE'
      END                          /* END IF OF COMPATABILITY CHECK   */
   END              /* END FOR 'CREATE TABLESPACE' CLAUSE CHECK     */

/*-----------------------------------------------------------------*/
/*  PARAGRAPH : REVIEW THAT CREATE INDEX USES THE CORRECT KEYWORDS   */
/*-----------------------------------------------------------------*/
IF DDLFUNC = 'CREATE' & OBJECT = 'INDEX' THEN DO
   CLAUSE_NAME = ''
   IF ( CNT_ROSHARE   > Ø ) THEN CLAUSE_NAME = 'ROSHARE'
   IF ( CNT_CCSID     > Ø ) THEN CLAUSE_NAME = 'CCSID'
   IF ( CNT_LARGE     > Ø ) THEN CLAUSE_NAME = 'LARGE'
   IF ( CNT_NUMPARTS  > Ø ) THEN CLAUSE_NAME = 'NUMPARTS'
   IF ( CNT_WORKFILE  > Ø ) THEN CLAUSE_NAME = 'WORKFILE'
   IF ( CNT_SEGSIZE   > Ø ) THEN CLAUSE_NAME = 'SEGSIZE'
   IF ( CNT_LOCKSIZE  > Ø ) THEN CLAUSE_NAME = 'LOCKSIZE'
   IF ( CNT_LOCKMAX   > Ø ) THEN CLAUSE_NAME = 'LOCKMAX'
   IF ( CNT_LOCKPART  > Ø ) THEN CLAUSE_NAME = 'LOCKPART'
   IF ( CNT_MAXROWS   > Ø ) THEN CLAUSE_NAME = 'MAXROWS'
   IF ( CLAUSE_NAME \= '' ) THEN DO
      SM = 'INVALID ' || CLAUSE_NAME
      LM = 'INVALID ' || CLAUSE_NAME || ,
           ' CLAUSE SPECIFIED IN CREATE INDEX STATEMENT'
      CALL DDLERROR YVAL, CL1, OBJECT, CLAUSE_NAME
      END
   IF ( DB2VER = 'V4' ) & ( CNT_PIECESIZE > Ø ) THEN DO
      LM = 'V5 CLAUSE ON DB2 V4 DDL.  MODIFY DDL OR PANEL VALUE'
      SM = 'USING V5 CLAUSE'
```

```
      CALL DDLERROR YVAL, CL1, DDLFUNC, 'PIECESIZE'
      END
   IF ( PARTFLAG = 'YES' ) & ( CNT_PIECESIZE > Ø ) THEN DO
      LM = 'DDL ERROR # 119: ',
           'PIECESIZE IS ONLY VALID FOR NON-PARTITIONED INDEXES'
      SM = 'DDL ERROR # 119'
      CALL DDLERROR YVAL, CL1, 'CLUSTER', 'PIECESIZE'
      END
   IF ( CNT_TYPE>Ø ) & ( INDXTYPE=2 ) & ( CNT_SUBPAGES>Ø ) THEN DO
      LM = 'SUBPAGES CLAUSE INVALID FOR INDEX TYPE 2'
      SM = 'CLAUSE ERROR'
      CALL DDLERROR YVAL, CL1, 'TYPE', 'SUBPAGES'
      END
   IF ( CNT_TYPE>Ø ) & ( INDXTYPE=1 ) & ( CNT_SUBPAGES=Ø ) THEN DO
      LM = 'SUBPAGES CLAUSE MUST BE SPECIFIED FOR INDEX TYPE 1'
      SM = 'CLAUSE ERROR'
      CALL DDLERROR YVAL, CL1, 'TYPE', '1'
      END
   IF ( CNT_ON_TABLE \= 1 ) THEN DO
      LM = 'DDL ERROR # 12Ø: MISSING THE "ON TABLE" CLAUSE'
      SM = 'DDL ERROR # 12Ø'
      CALL DDLERROR YVAL, CL1, DDLFUNC, OBJECT
      END
   END              /* END FOR 'CREATE INDEX' CLAUSES CHECK         */

/*-------------------------------------------------------------------*/
/*  PARAGRAPH : REVIEW THAT ALTER DATABASE USES CORRECT KEYWORDS.    */
/*-------------------------------------------------------------------*/
IF DDLFUNC = 'ALTER' & OBJECT = 'DATABASE' THEN DO
   CLAUSE_NAME = ''
   IF ( CNT_CCSID     > Ø ) THEN CLAUSE_NAME = 'CCSID'
   IF ( CNT_CLOSE     > Ø ) THEN CLAUSE_NAME = 'CLOSE'
   IF ( CNT_CLUSTER   > Ø ) THEN CLAUSE_NAME = 'CLUSTER'
   IF ( CNT_CONVERT   > Ø ) THEN CLAUSE_NAME = 'CONVERT'
   IF ( CNT_DEFER     > Ø ) THEN CLAUSE_NAME = 'DEFER'
   IF ( CNT_DSETPASS  > Ø ) THEN CLAUSE_NAME = 'DSETPASS'
   IF ( CNT_FREEPAGE  > Ø ) THEN CLAUSE_NAME = 'FREEPAGE'
   IF ( CNT_GBPCACHE  > Ø ) THEN CLAUSE_NAME = 'GBPCACHE'
   IF ( CNT_IN_DB     > Ø ) THEN CLAUSE_NAME = 'IN'
   IF ( CNT_LARGE     > Ø ) THEN CLAUSE_NAME = 'LARGE'
   IF ( CNT_LOCKMAX   > Ø ) THEN CLAUSE_NAME = 'LOCKMAX'
   IF ( CNT_LOCKPART  > Ø ) THEN CLAUSE_NAME = 'LOCKPART'
   IF ( CNT_LOCKSIZE  > Ø ) THEN CLAUSE_NAME = 'LOCKSIZE'
   IF ( CNT_MAXROWS   > Ø ) THEN CLAUSE_NAME = 'MAXROWS'
   IF ( CNT_NUMPARTS  > Ø ) THEN CLAUSE_NAME = 'NUMPARTS'
   IF ( CNT_ON_TABLE  > Ø ) THEN CLAUSE_NAME = 'ON'
   IF ( CNT_PART      > Ø ) THEN CLAUSE_NAME = 'PART'
   IF ( CNT_PCTFREE   > Ø ) THEN CLAUSE_NAME = 'PCTFREE'
   IF ( CNT_PIECESIZE > Ø ) THEN CLAUSE_NAME = 'PIECESIZE'
   IF ( CNT_PRIQTY    > Ø ) THEN CLAUSE_NAME = 'PRIQTY'
```

```
   IF ( CNT_SECQTY    > Ø ) THEN CLAUSE_NAME = 'SECQTY'
   IF ( CNT_SEGSIZE   > Ø ) THEN CLAUSE_NAME = 'SEGSIZE'
   IF ( CNT_SUBPAGES  > Ø ) THEN CLAUSE_NAME = 'SUBPAGES'
   IF ( CNT_TYPE      > Ø ) THEN CLAUSE_NAME = 'TYPE'
   IF ( CNT_UNIQUE    > Ø ) THEN CLAUSE_NAME = 'UNIQUE'
   IF ( CNT_WORKFILE  > Ø ) THEN CLAUSE_NAME = 'WORKFILE'
   IF ( CLAUSE_NAME \= '' ) THEN DO
      SM = 'INVALID ' || CLAUSE_NAME
      LM = 'INVALID ' || CLAUSE_NAME || ,
           ' CLAUSE SPECIFIED IN ALTER DATABASE STATEMENT'
      CALL DDLERROR YVAL, CL1, OBJECT, CLAUSE_NAME
      END
   END              /* END FOR 'ALTER DATABASE' CLAUSES CHECK       */

/*-------------------------------------------------------------------*/
/*  PARAGRAPH : REVIEW THAT ALTER TABLESPACE USES CORRECT KEYWORDS   */
/*-------------------------------------------------------------------*/
IF DDLFUNC = 'ALTER' & OBJECT = 'TABLESPACE' THEN DO
   CLAUSE_NAME = ''
   IF ( CNT_IN_DB     > Ø ) THEN CLAUSE_NAME = 'IN'
   IF ( CNT_ON_TABLE  > Ø ) THEN CLAUSE_NAME = 'ON'
   IF ( CNT_ROSHARE   > Ø ) THEN CLAUSE_NAME = 'ROSHARE'
   IF ( CNT_SUBPAGES  > Ø ) THEN CLAUSE_NAME = 'SUBPAGES'
   IF ( CNT_LARGE     > Ø ) THEN CLAUSE_NAME = 'LARGE'
   IF ( CNT_NUMPARTS  > Ø ) THEN CLAUSE_NAME = 'NUMPARTS'
   IF ( CNT_PIECESIZE > Ø ) THEN CLAUSE_NAME = 'PIECESIZE'
   IF ( CNT_UNIQUE    > Ø ) THEN CLAUSE_NAME = 'UNIQUE'
   IF ( CNT_CLUSTER   > Ø ) THEN CLAUSE_NAME = 'CLUSTER'
   IF ( CNT_SEGSIZE   > Ø ) THEN CLAUSE_NAME = 'SEGSIZE'
   IF ( CNT_DEFER     > Ø ) THEN CLAUSE_NAME = 'DEFER'
   IF ( CNT_TYPE      > Ø ) THEN CLAUSE_NAME = 'TYPE'
   IF ( CNT_WORKFILE  > Ø ) THEN CLAUSE_NAME = 'WORKFILE'
   IF ( CNT_CCSID     > Ø ) THEN CLAUSE_NAME = 'CCSID'
   IF ( CLAUSE_NAME \= '' ) THEN DO
      SM = 'INVALID ' || CLAUSE_NAME
      LM = 'INVALID ' || CLAUSE_NAME || ,
           ' CLAUSE SPECIFIED IN ALTER TABLESPACE STATEMENT'
      CALL DDLERROR YVAL, CL1, OBJECT, CLAUSE_NAME
      END
   IF ( DB2VER = 'V4' ) THEN DO
      CLAUSE_NAME = ''
      IF ( CNT_MAXROWS  > Ø ) THEN CLAUSE_NAME = 'MAXROWS'
      IF ( CNT_LOCKPART > Ø ) THEN CLAUSE_NAME = 'LOCKPART'
      IF ( CLAUSE_NAME \= '' ) THEN DO
         LM = 'V5 CLAUSE ON DB2 V4 DDL.  MODIFY DDL OR PANEL VALUE'
         SM = 'V5 CLAUSE>'||CLAUSE_NAME
         CALL DDLERROR YVAL, CL1, DDLFUNC, CLAUSE_NAME
         END
      END            /* END FOR IF DB2VER = 'V4' CHECK               */
   /* CHECK COMPATABILITY OF LOCKPART AND LOCKSIZE PARAMETERS (V5)   */
```

```
   IF ( ( CNT_LOCKPART > Ø ) & ( CNT_LOCKSIZE > Ø ) ) THEN DO
      IF ( ( LOCKPART_PARM = 'YES' ) & ,
           ( LOCKSIZE_PARM = 'TABLESPACE') ) THEN DO
         LM = '"LOCKPART YES" AND "LOCKSIZE TABLESPACE" NOT COMPATIBLE'
         SM = 'INCOMPATIBLE PARMS'
         CALL DDLERROR YVAL, CL1, DDLFUNC, 'LOCKPART'
         END
      END                          /* END IF OF COMPATABILITY CHECK   */
   END              /* END FOR 'ALTER TABLESPACE' CLAUSES CHECK     */

/*-----------------------------------------------------------------*/
/*  PARAGRAPH : REVIEW THAT ALTER INDEX USES THE CORRECT KEYWORDS  */
/*-----------------------------------------------------------------*/
IF DDLFUNC = 'ALTER' & OBJECT = 'INDEX' THEN DO
   CLAUSE_NAME = ''
   IF ( CNT_IN_DB     > Ø ) THEN CLAUSE_NAME = 'IN'
   IF ( CNT_ON_TABLE  > Ø ) THEN CLAUSE_NAME = 'ON'
   IF ( CNT_ROSHARE   > Ø ) THEN CLAUSE_NAME = 'ROSHARE'
   IF ( CNT_LARGE     > Ø ) THEN CLAUSE_NAME = 'LARGE'
   IF ( CNT_NUMPARTS  > Ø ) THEN CLAUSE_NAME = 'NUMPARTS'
   IF ( CNT_MAXROWS   > Ø ) THEN CLAUSE_NAME = 'MAXROWS'
   IF ( CNT_UNIQUE    > Ø ) THEN CLAUSE_NAME = 'UNIQUE'
   IF ( CNT_CLUSTER   > Ø ) THEN CLAUSE_NAME = 'CLUSTER'
   IF ( CNT_SEGSIZE   > Ø ) THEN CLAUSE_NAME = 'SEGSIZE'
   IF ( CNT_DEFER     > Ø ) THEN CLAUSE_NAME = 'DEFER'
   IF ( CNT_WORKFILE  > Ø ) THEN CLAUSE_NAME = 'WORKFILE'
   IF ( CNT_CCSID     > Ø ) THEN CLAUSE_NAME = 'CCSID'
   IF ( CNT_LOCKPART  > Ø ) THEN CLAUSE_NAME = 'LOCKPART'
   IF ( CLAUSE_NAME \= '' ) THEN DO
      SM = 'INVALID ' || CLAUSE_NAME
      LM = 'INVALID ' || CLAUSE_NAME || ,
           ' CLAUSE SPECIFIED IN ALTER INDEX STATEMENT'
      CALL DDLERROR YVAL, CL1, OBJECT, CLAUSE_NAME
      END
   IF ( DB2VER = 'V4' ) & ( CNT_PIECESIZE > Ø ) THEN DO
      LM = 'V5 CLAUSE ON DB2 V4 DDL.  MODIFY DDL OR PANEL VALUE'
      SM = 'USING V5 CLAUSE'
      CALL DDLERROR YVAL, CL1, DDLFUNC, OBJECT
      END
   IF ( CNT_TYPE>Ø ) & ( INDXTYPE=2 ) & ( CNT_SUBPAGES>Ø ) THEN DO
      LM = 'SUBPAGES CLAUSE INVALID FOR INDEX TYPE 2'
      SM = 'CLAUSE ERROR'
      CALL DDLERROR YVAL, CL1, 'TYPE', 'SUBPAGES'
      END
   END              /* END FOR 'ALTER INDEX' CLAUSES CHECK          */
RETURN              /* EXIT FOR THE CHECK_CLAUSES SUBROUTINE.       */

/*****************************************************************/
/*  SUB-ROUTINE CHECK_USING_PARM:                                */
/*  THIS SUB-ROUTINE VALIDATE THAT THE STORAGE TYPE AND STORAGE NAME */
```

```
/*  FIELDS IN THE DDL ARE EQUAL TO THE USER SPECIFIED VALUES.        */
/*********************************************************************/
/*  PARM START_LINE INDICATES WHERE TO POSITION CURSOR TO FIND FROM. */
/*  PARM OBJECT_TYPE INDICATES WHAT USER DEFAULTS SHOULD BE USED     */
/*  PARM STORTYPE CONTAINS THE STOGROUP OR VCAT PARMS OF USING CLAUSE*/
/*  PARM STORNAME CONTAINS THE STOGROUP NAME OR VCAT NAME THAT IS    */
/*       BEING CHECKED AGAINST THE USER DEFAULT VALUES SPECIFIED.    */
/*  PARM KEYWORDCNT HAS THE # OF OCCURRENCES OF KEYWORD TO LOOK FOR. */
/*********************************************************************/
CHECK_USING_PARM: PROCEDURE EXPOSE CL1 SM LM STOPFLAG YVAL,
         DFTSSTOR DFTSNAME DFIXSTOR DFIXNAME SETDFLAG DDLSTR
ARG START_LINE, OBJECT_TYPE, STORTYPE, STORNAME, KEYWORDCNT
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE CHECK_USING_PARM BEGINS HERE.                        */
/*-------------------------------------------------------------------*/
IF ( KEYWORDCNT = '' ) THEN KEYWORDCNT = 1      /* CHECK IF NULL VAL */
IF ( DATATYPE(KEYWORDCNT,'NUM') \= 1 ) THEN DO
   LM = 'ERROR ON CHECK_USING_PARM ROUTINE. GOT A NON NUMERIC VALUE'
   SM = 'EDIT MACRO ERROR'
   'ISPEXEC SETMSG MSG(ISRZØØ1)'                /* SET ISPF MSG VARS */
   EXIT 8                                       /* ERROR IN THE PGM  */
   END                                          /* EXIT MACRO        */
/*-------------------------------------------------------------------*/
/*  SET DEFAULT VARIABLES FOR OBJECT TYPE WITH USER DEFAULTS         */
/*-------------------------------------------------------------------*/
IF OBJECT_TYPE = 'TS' THEN DO   /* SET USER DEFAULTS FOR TABLESPACES */
   DEF_STOR_TYPE = DFTSSTOR
   DEF_STOR_NAME = DFTSNAME
   END
IF OBJECT_TYPE = 'IX' THEN DO   /* SET USER DEFAULTS FOR INDEXSPACES */
   DEF_STOR_TYPE = DFIXSTOR
   DEF_STOR_NAME = DFIXNAME
   END
/*-------------------------------------------------------------------*/
/*  VALIDATE THAT DEF_STOR_NAME IS A VALID NAME BEFORE UPDATES.      */
/*-------------------------------------------------------------------*/
CALL CHECK_VALID_NAME STORNAME, STORTYPE           /* VALIDATE NAME */
/*-------------------------------------------------------------------*/
/*  CHECK IF STORAGE VALUE IS THE SAME AS THE SPECIFIED USER DEFAULT */
/*  AND ISSUE NOTE MESSAGES WHENEVER THERE ARE CHANGES IN STOR_TYPE. */
/*-------------------------------------------------------------------*/
IF ( STORTYPE \= DEF_STOR_TYPE ) THEN DO   /* CHECK AGAINST USER DEF */
   'ISREDIT CURSOR = ' START_LINE Ø
   'ISREDIT FIND "'STORTYPE'" NEXT WORD'
   'ISREDIT (XL1,XC1) = CURSOR'
   IF ( SETDFLAG = YVAL ) THEN DO          /* ENFORCE DEFAULTS FLAG  */
      'ISREDIT (LNX) = LINE' XL1
      PARSE VAR LNX F_TXT (STORTYPE) E_TXT
      LNX = F_TXT DEF_STOR_TYPE E_TXT
      CALL INSERT_LINE 'SAME', XL1, LNX
```

```
      NT = 'STORAGE TYPE' STORTYPE 'CHANGED TO DEFAULT' DEF_STOR_TYPE
      CALL WRITE_MSG NT, XL1
      IF ( STORTYPE = 'VCAT' ) THEN DO     /* FROM VCAT TO STOGROUP  */
         NT = 'CHECK DDL WHEN CONVERTING FROM "VCAT" TO "STOGROUP"'
         CALL WRITE_NOTE NT, XL1
         NT = 'CHECK THAT PRIQTY AMOUNT IS SPECIFIED CORRECTLY'
         CALL WRITE_NOTE NT, XL1
         END
      ELSE DO                              /* FROM STOGROUP TO VCAT  */
         NT = 'CHECK DDL WHEN CONVERTING FROM "STOGROUP" TO "VCAT"'
         CALL WRITE_NOTE NT, XL1
         NT = 'PRIQTY & SECQTY CLAUSES ARE NOT NEEDED WITH "VCAT"'
         CALL WRITE_NOTE NT, XL1
         END
      END
   ELSE DO                                 /* DO NOT ENFORCE DFLTS   */
      NT = 'NOT USING DEFAULT STORAGE TYPE ' DEF_STOR_TYPE
      CALL WRITE_NOTE NT, XL1
      END
   END
/*-------------------------------------------------------------------*/
/*  CHECK IF STORAGE NAME IS SAME AS USER DEFAULT SPECIFIED. ISSUE   */
/*  MESSAGES AS APPROPRIATE.                                         */
/*-------------------------------------------------------------------*/
IF ( STORNAME \= DEF_STOR_NAME ) THEN DO   /* CHECK AGAINST USER DEF */
   'ISREDIT CURSOR = ' START_LINE Ø
   'ISREDIT FIND "'STORNAME'" NEXT WORD'
   'ISREDIT (XL1,XC1) = CURSOR'
   IF ( SETDFLAG = YVAL ) THEN DO          /* ENFORCE DEFAULTS FLAG  */
      'ISREDIT (LNX) = LINE' XL1
      /* CHECK TO SEE IF (DEF_STOR_TYPE) IS IN THE SAME LINE         */
      IF (WORDPOS((DEF_STOR_TYPE),LNX) > Ø ) THEN DO
         PARSE VAR LNX F_TXT (DEF_STOR_TYPE) (STORNAME) E_TXT
         LNX = F_TXT DEF_STOR_TYPE DEF_STOR_NAME
         END
      ELSE DO                          /* NO, IT IS NOT IN SAME LINE */
         PARSE VAR LNX F_TXT (STORNAME) E_TXT
         LNX = F_TXT DEF_STOR_NAME E_TXT
         END
      CALL INSERT_LINE 'SAME', XL1, LNX
      NT = 'STORAGE NAME' STORNAME 'CHANGED TO DEFAULT' DEF_STOR_NAME
      CALL WRITE_MSG NT, XL1
      END
   ELSE DO                                 /* DO NOT ENFORCE DFLTS   */
      NT = 'NOT USING DEFAULT' STORTYPE 'NAME' DEF_STOR_NAME
      CALL WRITE_NOTE NT, XL1
      END
   END
RETURN                     /* END OF SUB-ROUTINE CHECK_USING_PARM */
```

```
/*********************************************************************/
/*  SUB-ROUTINE CHECK_BUFFERPOOL_PARM:                               */
/*  VERIFY THAT BUFFERPOOL NAME IS VALID BUFFERPOOL NAME             */
/*  IN-PARM   BPNAME : CONTAINS THE BPNAME BEING VALIDATED.          */
/*  OUT-PARM  TAIL_SEMI_COLON : CONTAINS A '' OR A ';' CHARACTER.    */
/*********************************************************************/
CHECK_BUFFERPOOL_PARM: PROCEDURE EXPOSE CL1 DDLSTR,
                       YVAL STOPFLAG
ARG BPNAME
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE CHECK_BUFFERPOOL_PARM BEGINS HERE                    */
/*-------------------------------------------------------------------*/
/*  CHECK IF THE BPNAME HAS A ';' IMMEDIATLY FOLLOWING THE NAME.     */
/*  IF SO, THEN STRIP IT AND SET VARIABLE TAIL_SEMI_COLON TO ';'.    */
/*-------------------------------------------------------------------*/
TBPNAME = STRIP(BPNAME,T,';')
BPNMLEN = LENGTH(TBPNAME)
IF (BPNMLEN < 3) | ( BPNMLEN > 6) THEN DO  /* CHECK LENGTH OF BPNAME */
   LM = 'INCORRECT BUFFERPOOL NAME'
   SM = 'INCORRECT BPNAME'
   CALL DDLERROR YVAL, CL1, 'BUFFERPOOL', TBPNAME, 1
   END
IF ( BPNMLEN <= 4) &,
   (SUBSTR(TBPNAME,1,2) = 'BP') THEN DO      /* CHECK FOR 4K BPNAME  */
   BPNUM = SUBSTR(TBPNAME,3)
   IF ( DATATYPE(BPNUM,'NUM') \= 1 ) THEN DO
      LM = 'INCORRECT BUFFERPOOL NAME -' TBPNAME
      SM = 'INCORRECT BPNAME'
      CALL DDLERROR YVAL, CL1, 'BUFFERPOOL', TBPNAME, 1
      END
   IF ( BPNUM > 49 ) THEN DO
      LM = 'INCORRECT BUFFERPOOL NUMBER -' TBPNAME
      SM = 'INCORRECT BPNAME'
      CALL DDLERROR YVAL, CL1, 'BUFFERPOOL', TBPNAME,1
      END
   END                                       /* END FOR 4K BPNAME    */
ELSE DO                                      /* CHECK FOR 32K BPNAME */
   IF ( BPNMLEN > 6 ) THEN DO
      LM = 'INCORRECT BUFFERPOOL NAME - ' TBPNAME
      SM = 'INCORRECT BPNAME'
      CALL DDLERROR YVAL, CL1, 'BUFFERPOOL', TBPNAME, 1
      END
   IF ( BPNMLEN = 5 ) THEN DO
      IF (TBPNAME \= 'BP32K') THEN DO
         LM = 'INCORRECT BUFFERPOOL NAME'
         SM = 'INCORRECT BPNAME'
         CALL DDLERROR YVAL, CL1, 'BUFFERPOOL', TBPNAME, 1
         END
      END
   IF ( BPNMLEN = 6 ) THEN DO
```

```
      IF (SUBSTR(TBPNAME,1,5) = 'BP32K') THEN DO
         BPNUM = SUBSTR(TBPNAME,6)
         IF ( DATATYPE(BPNUM,'NUM') \= 1 ) THEN DO
            LM = 'VALID BPNAMES ARE BP32K, BP32K1 .. BP32K9'
            SM = 'INCORRECT BPNAME'
            CALL DDLERROR YVAL, CL1, 'BUFFERPOOL', TBPNAME, 1
            END
         IF (BPNUM > 9) THEN DO
            LM = 'VALID BPNAMES ARE BPK32, BP32K1 .. BP32K9'
            SM = 'INCORRECT BPNAME'
            CALL DDLERROR YVAL, CL1, 'BUFFERPOOL', TBPNAME, 1
            END
         END
      END
   END                                        /* END FOR 32K BPNAME   */
RETURN                        /* END SUB-ROUTINE PARSE_BUFFERPOOL PARM */

/*********************************************************************/
/*  SUB-ROUTINE CHECK_SEGSIZE_PARM:                                  */
/*  ALL OF THE VALUES NEEDED BY THIS SUB-ROUTINE ARE EXPOSED TO IT.  */
/*********************************************************************/
CHECK_SEGSIZE_PARM: PROCEDURE EXPOSE CL1 DDLSTR CURPOS YVAL,
             DB2VER SEGMPOS SEGSZ STOPFLAG
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE CHECK_SEGSIZE_PARM BEGINS HERE                       */
/*-------------------------------------------------------------------*/
/*  CHECK IF SEGSIZE HAS A ';' IMMEDIATLY FOLLOWING IT.              */
/*  IF SO, THEN STRIP IT AND SET VARIABLE TAIL_SEMI_COLON TO ';'.    */
/*-------------------------------------------------------------------*/
SEGSZ = WORD(DDLSTR,(SEGMPOS+1))
SEGSZ = STRIP(SEGSZ,T,';')
/*  ENSURE SEGMENT SIZE IS A VALID NUMBER                            */
IF ( DATATYPE(SEGSZ,'NUM') \= 1 ) THEN DO
   LM = 'DDL ERROR # 121: SEGMENT SIZE NOT NUMERIC ='||SEGSZ
   SM = 'DDL ERROR # 121'
   CALL DDLERROR STOPFLAG, CL1, 'SEGSIZE', SEGSZ, 1
   END
/*  ENSURE SEGMENT SIZE IS IN THE VALID RANGE (4 TO 64)              */
IF ( SEGSZ < 4 ) | ( SEGSZ  > 64 ) THEN DO
   LM = 'DDL ERROR # 122: SEGMENT SIZE MUST BE >=4 & <=64'
   SM = 'DDL ERROR # 122'
   CALL DDLERROR YVAL, CL1, 'SEGSIZE', SEGSZ, 1
   END
/*  ENSURE SEG-SIZE IS AN EVEN MULTIPLE OF FOUR (4)                  */
IF ( ( SEGSZ // 4 ) \= Ø ) THEN DO
   LM = 'DDL ERROR # 123: SEGMENT SIZE MUST BE A MULTIPLE OF 4'
   SM = 'DDL ERROR # 123'
   CALL DDLERROR YVAL, CL1, 'SEGSIZE', '', 1
   END
RETURN                     /* END OF SUB-ROUTINE CHECK_SEGSIZE_PARM */
```

```
/*********************************************************************/
/*  SUB-ROUTINE CHECK_NUMPARTS_PARM:                                 */
/*  ONLY NUMPARTS PARM IS USED.   ALL OF THE VALUES NEEDED BY THIS   */
/*  SUB-ROUTINE ARE EXPOSED TO IT.                                   */
/*********************************************************************/
CHECK_NUMPARTS_PARM: PROCEDURE EXPOSE CL1 DDLSTR CURPOS,
                     YVAL DB2VER
ARG NUMPARTS
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE CHECK_NUMPARTS_PARM BEGINS HERE                      */
/*-------------------------------------------------------------------*/
/*  PARAGRAPH :  PARSE/VALIDATE NUMPARTS CLAUSE.  NO PARMS ARE PASSED*/
/*  VALIDATE THAT NUMPARTS IS NUMERIC AND APPROPIATE FOR DB2 VERSION */
/*-------------------------------------------------------------------*/
IF ( DATATYPE(NUMPARTS,'NUM') \= 1 ) THEN DO
   LM = 'DDL ERROR # 124: NUMBER OF PARTS NOT A NUMERIC VALUE'
   SM = 'DDL ERROR # 124'
   CALL DDLERROR YVAL, CL1, 'NUMPARTS', '', 1
   END
/* VALIDATE THAT NUMPARTS FOR DB2 V4 IS IN THE RANGE OF 1 TO 64.     */
IF (DB2VER = 'V4') THEN DO
   IF ( ( NUMPARTS < 1 ) | ( NUMPARTS > 64 ) ) THEN DO
      LM = 'DDL ERROR # 125: DB2 V4 - NUMBER OF PARTS < 1 OR > 64 '
      SM = 'DDL ERROR # 125'
      CALL DDLERROR YVAL, CL1, NUMPARTS, '', 1
      END
   END
/* VALIDATE THAT NUMPARTS FOR DB2 V5 IS IN THE RANGE OF 1 TO 255.    */
IF (DB2VER = 'V5') THEN DO
   IF ( ( NUMPARTS < 1 ) | ( NUMPARTS > 255 ) ) THEN DO
      LM = 'DDL ERROR # 126: DB2 V5 - NUMBER OF PARTS < 1 OR > 255'
      SM = 'DDL ERROR # 126'
      CALL DDLERROR YVAL, CL1, NUMPARTS, '', 1
      END
   END
RETURN                   /* END OF SUB-ROUTINE CHECK_NUMPARTS_PARM*/

/*********************************************************************/
/*  SUB-ROUTINE CHECK_PARTNO_PARM:                                   */
/*  NO PARAMETERS ARE SPECIFIED.  ALL OF THE VALUES NEEDED BY THIS   */
/*  SUB-ROUTINE ARE EXPOSED TO IT.                                   */
/*********************************************************************/
CHECK_PARTNO_PARM: PROCEDURE EXPOSE CL1 DDLSTR CURPOS,
       CNTPART PARTNO NUMPARTS STOPFLAG YVAL
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE CHECK_PARTNO_PARM BEGINS HERE.                       */
/*-------------------------------------------------------------------*/
PARTNO = WORD(DDLSTR,(CURPOS+1))
CNTPART = CNTPART + 1
```

```
IF ( DATATYPE(PARTNO,'NUM') \= 1 ) THEN DO
   LM = 'DDL ERROR # 127: COULD NOT GET THE PART NUMBER'
   SM = 'DDL ERROR # 127'
   CALL DDLERROR YVAL, CL1, 'PART', PARTNO, CNTPART
   END
IF ( PARTNO > NUMPARTS ) THEN DO
   LM = 'DDL ERROR # 128: PART NUMBER >   NUMPARTS'
   SM = 'DDL ERROR # 128'
   CALL DDLERROR YVAL, CL1, 'PART', PARTNO, CNTPART
   END
RETURN                     /* END OF SUB-ROUTINE CHECK_PARTNO_PARM */

/*********************************************************************/
/*   SUB-ROUTINE : CHECK_INDEXTYPE_PARM:    NO ARGUMENTS ARE PASSED. */
/*   THIS SUB-ROUTINE CHECKS THAT THE INDEXTYPE PARAMETER SPECIFIED  */
/*   IS A VALID NUMBER AND HAS A VALUE OF 1 OR 2.                    */
/*********************************************************************/
CHECK_INDEXTYPE_PARM: PROCEDURE EXPOSE CL1 DB2VER DDLSTR,
                        INDXTYPE STOPFLAG   YVAL
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE CHECK_INDEXTYPE_PARM BEGINS HERE.                    */
/*-------------------------------------------------------------------*/
/*  PARAGRAPH : CHECK THAT INDXTYPE IS A VALID NUMBER.               */
/*-------------------------------------------------------------------*/
IF ( DATATYPE(INDXTYPE,'NUM') \= 1 ) THEN DO
   LM = 'DDL ERROR # 129: INDEXTYPE PARAMETER MUST BE A NUMERIC VALUE'
   SM = 'DDL ERROR # 129'
   CALL DDLERROR YVAL, CL1, 'INDEXTYPE', INDXTYPE, 1
   END
IF \ ( ( INDXTYPE = 1 ) | ( INDXTYPE = 2) ) THEN DO
   LM = 'DDL ERROR # 130: INDEXTYPE MUST BE A 1 OR A 2'
   SM = 'DDL ERROR # 130'
   CALL DDLERROR YVAL, CL1, 'INDEXTYPE', INDXTYPE, 1
   END
RETURN                      /* END OF SUB-ROUTINE CHECK_INDEXTYPE    */

/*********************************************************************/
/*   SUB-ROUTINE : CHECK_INDEX_PARTS.       NO ARGUMENTS ARE PASSED. */
/*   AFTER PARSING THE INDEX PARTS, THIS SUB-ROUTINE CHECKS THAT ALL */
/*   PARTS FOR A PARTITIONED INDEX ARE DEFINED.  THIS IS DONE BY     */
/*   CHECKING THAT THE VALUES FIELD IS NOT EQUAL TO 'ERROR'          */
/*********************************************************************/
CHECK_INDEX_PARTS: PROCEDURE EXPOSE CL1 DB2VER DDLSTR,
          NUMPARTS     STOPFLAG   YVAL,
          HIGHEST_PARTNO          P_VALUES.
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE CHECK_INDEX_PARTS BEGINS HERE.                       */
/*-------------------------------------------------------------------*/
/*  PARAGRAPH : LOOP THRU THE ARRAY OF INDEX PARTS LOOKING FOR ERROR */
/*-------------------------------------------------------------------*/
```



```
DO PARTNO=1 TO NUMPARTS BY 1
   IF P_VALUES.PARTNO = 'ERROR' THEN DO
      LM = 'DDL ERROR # 131: SPECIFICATIONS FOR PART ' || PARTNO || ,
           ' WERE NOT FOUND'
      SM = 'DDL ERROR # 131'
      CALL DDLERROR YVAL, CL1, CLUSTER
      END
   END
RETURN                      /* END OF SUB-ROUTINE CHECK_INDEX_PARTS  */

/*********************************************************************/
/*   SUB-ROUTINE : CHECK_VALID_NAME.   TWO PARAMETERS ARE PASSED.    */
/*   THIS ROUTINE CHECKS THAT THE 1ST PARM  PASSED IS A STRING, WITH */
/*   THE FIRST CHARACTER BEING ALPHA AND WITH LESS THAN 8 CHARS.     */
/*   THE SECOND PARAMETER IS USED FOR POSSITIONING THE CURSOR.       */
/*********************************************************************/
CHECK_VALID_NAME: PROCEDURE EXPOSE CL1 DDLSTR STOPFLAG YVAL
ARG NAME_TO_CHECK, WORD_MARKER
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE CHECK_VALID_NAME  BEGINS HERE.                       */
/*-------------------------------------------------------------------*/
STRIP_NAME = STRIP(NAME_TO_CHECK,'B',';')
NWORDS  = WORDS(STRIP_NAME)                    /* COUNT THE # WORDS  */
IF ( NWORDS \= 1 ) THEN DO                     /* CHECK THE COUNT    */
   LM = 'DDL ERROR # 132: EXPECTING ONLY ONE STRING, FOUND 'STRIP_NAME
   SM = 'DDL ERROR # 132'
   CALL DDLERROR YVAL, CL1, WORD_MARKER, STRIP_NAME, 1
   END
IF ( DATATYPE(STRIP_NAME,'A') = 0 ) THEN DO    /* CHECK THE TYPE     */
   LM = 'DDL ERROR # 133: NAME' STRIP_NAME 'IS NOT A VALID STRING.'
   SM = 'DDL ERROR # 133'
   CALL DDLERROR YVAL, CL1, WORD_MARKER, STRIP_NAME, 1
   END
TOKENLEN = LENGTH(STRIP_NAME)                  /* GET THE LENGTH     */
IF ( TOKENLEN < 1 ) | ( TOKENLEN > 8 ) THEN DO /* CHECK THE LENGTH   */
   LM = 'DDL ERROR # 134: LENGTH OF NAME MUST BE 1 TO 8'
   SM = 'DDL ERROR # 134'
   CALL DDLERROR YVAL, CL1, WORD_MARKER, STRIP_NAME, 1
   END
FIRST_CHAR = SUBSTR(STRIP_NAME,1,1)            /* GET 1 CHAR IN STR  */
IF ( DATATYPE(FIRST_CHAR,'U') = 0 ) THEN DO    /* CHECK IF UP-ALPHA  */
   LM = 'DDL ERROR # 135: FIRST CHARACTER IN NAME MUST BE A LETTER'
   SM = 'DDL ERROR # 135'
   CALL DDLERROR YVAL, CL1, WORD_MARKER, STRIP_NAME, 1
   END
RETURN                      /* END OF SUB-ROUTINE CHECK_VALID_NAME   */

/*********************************************************************/
/*   SUB-ROUTINE : PARSE_TABLESPACE_PARTS.  NO ARGUMENTS ARE PASSED. */
/*   THIS SUB-ROUTINE WILL PARSE THE PARTITIONED TABLESPACES PARTS   */
```

```
/*   CLAUSES FOR ALL NON-SPACE RELATED CLAUSES.  ONCE A CLAUSE IS    */
/*   FOUND, A SUB-ROUTINE IS CALLED TO VALIDATE THE KEYWORDS FOR THE */
/*   FOUND CLAUSE.  CLAUSES AND KEYWORDS ARE STORED IN ARRAYS.       */
/*********************************************************************/
PARSE_TABLESPACE_PARTS: PROCEDURE EXPOSE CL1 DB2VER DDLSTR,
          NUMPARTS     NPRTPOS      PARTNO      STOPFLAG,
          DDLWORDS     QTY          CMPYN       GBPTKN,
          CURPOS       SC_HEX_VAL   YVAL        PARTNO,
          DFTSSTOR     DFTSNAME     DFIXSTOR    DFIXNAME,
          SPLITRW1     SPLITCL1     SPLITRW2    SPLITCL2,
          P_STOR_TYPE. P_STOR_NAME. P_PRIQTY.   P_SECQTY. ,
          P_N_PRIQTY.  P_N_SECQTY.  P_ERASE.    P_FREEPAGE. ,
          P_PCTFREE.   P_GBPCACHE.  P_COMPRESS. P_VALUES. ,
          SETDFLAG     P_SEGSIZE    P_N_SEGSIZE ,
          CNTPART      CNTUSING     CNTPRIQTY   CNTSECQTY ,
          CNTCOMPRESS  CNTERASE     CNTFREEPAGE CNTPCTFREE ,
          CNTGBPCACHE  CNTVALUES    U_STOR      U_NAME
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE PARSE_TABLESPACE_PART BEGINS HERE.                   */
/*-------------------------------------------------------------------*/
/*  PARAGRAPH : START PARSING 2 WORDS AFTER NUMPARTS CLAUSE.  CHECK  */
/*  THAT THE WORD IS A LEFT PARENTHESES.  REPOSITION CURSOR ON IT.   */
/*-------------------------------------------------------------------*/
BEGSPLIT = WORD(DDLSTR,(NPRTPOS+2))        /* SAVE BEGSPLIT WORD     */
SAVTOKEN = ''                              /* INITIALIZE VARIABLE    */
IF ( BEGSPLIT \= '(' ) THEN DO             /* CHECK IS A '(' WORD    */
   LM = 'DDL ERROR # 136: EXPECTING TO FIND A LEFT PARENTHESES'
   SM = 'DDL ERROR # 136'
   CALL DDLERROR YVAL, CL1, 'CLUSTER', BEGSPLIT, 1
   END
'ISREDIT CURSOR = ' CL1 Ø                  /* REPOSITION CURSOR      */
'ISREDIT FIND NUMPARTS WORD NEXT'          /* POSITION AT NUMPARTS   */
'ISREDIT FIND "(" WORD NEXT'               /* POSITION AT LPAREN     */
'ISREDIT (SPLITRW1,SPLITCL1) = CURSOR'     /* SAVE SPLIT LOCATION    */

/*-------------------------------------------------------------------*/
/*  PARAGRAPH : LOOP THRU THE DDL LOOKING FOR PARAMETERS FOR CLAUSES */
/*-------------------------------------------------------------------*/
CURPOS = NPRTPOS + 3                /* START PAST LEFT PARENTHESES   */
CNT_COMMA = Ø                       /* INITIALIZE COMMA COUNTER      */
HIGHEST_PARTNO = Ø                  /* INITIALIZE HIGHEST_PARTNO     */
DO WHILE ( CURPOS <= DDLWORDS )
   TOKEN = WORD(DDLSTR,CURPOS)      /* GET NEXT TOKEN IN DDL STRING  */
   IF ( TOKEN = ';' ) THEN ,
        'ISREDIT FIND X"'SC_HEX_VAL'" NEXT'
   ELSE 'ISREDIT FIND "'TOKEN'" NEXT WORD'
   SELECT
      WHEN TOKEN = ',' THEN DO      /* BEGIN OF WHEN STATEMENT       */
         CNT_COMMA = CNT_COMMA + 1           /* INCREASE COMMA CNT  */
         CURPOS = CURPOS + 1        /* INCREMENT CURSOR POINTER BY 2 */
```

```
      IF CNT_COMMA > HIGHEST_PARTNO THEN DO
         LM = 'DDL ERROR # 137: COMMA FOUND BEFORE PART CLAUSE'
         SM = 'DDL ERROR # 137'
         CALL DDLERROR YVAL, CL1, ',', '', CNT_COMMA
         END
      ITERATE                    /* DO THE LOOP AGAIN             */
      END                        /* END OF WHEN STATEMENT         */
   WHEN TOKEN = 'PART'  THEN DO  /* BEGIN OF WHEN STATEMENT       */
      CALL CHECK_PARTNO_PARM     /* CALL CHECKER SUB-ROUTINE      */
      IF PARTNO > HIGHEST_PARTNO THEN ,
         HIGHEST_PARTNO = PARTNO        /* SET HIGHEST PART NO    */
      CURPOS = CURPOS + 2        /* INCREMENT CURSOR POINTER BY 1 */
      ITERATE                    /* DO THE LOOP AGAIN             */
      END                        /* END OF WHEN STATEMENT         */
   WHEN TOKEN = 'USING' THEN DO  /* BEGIN OF WHEN STATEMENT       */
      CALL PARSE_USING_PARM      /* CALL CHECKER SUB-ROUTINE      */
      P_STOR_TYPE.PARTNO = U_STOR /* SAVE PARAMETER IN ARRAY.     */
      P_STOR_NAME.PARTNO = U_NAME /* SAVE PARAMETER IN ARRAY.     */
      CALL CHECK_VALID_NAME U_NAME, U_STOR   /* VALIDATE NAME     */
      CURPOS = CURPOS + 3        /* INCREMENT CURSOR POINTER BY 3 */
      ITERATE                    /* DO THE LOOP AGAIN             */
      END                        /* END OF WHEN STATEMENT         */
   WHEN TOKEN = 'PRIQTY' THEN DO /* BEGIN OF WHEN STATEMENT       */
      CALL PARSE_PRIQTY_PARM     /* CALL CHECKER SUB-ROUTINE      */
      P_PRIQTY.PARTNO   = QTY    /* SAVE PARAMETER IN ARRAY.      */
      CURPOS = CURPOS + 2        /* INCREMENT CURSOR POINTER BY 2 */
      ITERATE                    /* DO THE LOOP AGAIN             */
      END                        /* END OF WHEN STATEMENT         */
   WHEN TOKEN = 'SECQTY' THEN DO /* BEGIN OF WHEN STATEMENT       */
      CALL PARSE_SECQTY_PARM     /* CALL CHECKER SUB-ROUTINE      */
      P_SECQTY.PARTNO   = QTY    /* SAVE PARAMETER IN ARRAY.      */
      CURPOS = CURPOS + 2        /* INCREMENT CURSOR POINTER BY 2 */
      ITERATE                    /* DO THE LOOP AGAIN             */
      END                        /* END OF WHEN STATEMENT         */
   WHEN TOKEN = 'COMPRESS' THEN DO /* BEGIN OF WHEN STATEMENT     */
      CALL PARSE_COMPRESS_PARM   /* CALL CHECKER SUB-ROUTINE      */
      P_COMPRESS.PARTNO = CMPYN  /* SAVE PARAMETER IN ARRAY.      */
      CURPOS = CURPOS + 2        /* INCREMENT CURSOR POINTER BY 2 */
      ITERATE                    /* DO THE LOOP AGAIN             */
      END                        /* END OF WHEN STATEMENT         */
   WHEN TOKEN = 'ERASE' THEN DO  /* BEGIN OF WHEN STATEMENT       */
      CALL PARSE_ERASE_PARM      /* CALL CHECKER SUB-ROUTINE      */
      P_ERASE.PARTNO    = ERASE  /* SAVE PARAMETER IN ARRAY.      */
      CURPOS = CURPOS + 2        /* INCREMENT CURSOR POINTER BY 2 */
      ITERATE                    /* DO THE LOOP AGAIN             */
      END                        /* END OF WHEN STATEMENT         */
   WHEN TOKEN = 'FREEPAGE' THEN DO /* BEGIN OF WHEN STATEMENT     */
      CALL PARSE_FREEPAGE_PARM   /* CALL CHECKER SUB-ROUTINE      */
      P_FREEPAGE.PARTNO = FPG    /* SAVE PARAMETER IN ARRAY.      */
      CURPOS = CURPOS + 2        /* INCREMENT CURSOR POINTER BY 2 */
```

```
         ITERATE                    /* DO THE LOOP AGAIN            */
         END                        /* END OF WHEN STATEMENT        */
      WHEN TOKEN = 'PCTFREE' THEN DO /* BEGIN OF WHEN STATEMENT     */
         CALL PARSE_PCTFREE_PARM    /* CALL CHECKER SUB-ROUTINE     */
         P_PCTFREE.PARTNO  = PCT    /* SAVE PARAMETER IN ARRAY.     */
         CURPOS = CURPOS + 2        /* INCREMENT CURSOR POINTER BY 2 */
         ITERATE                    /* DO THE LOOP AGAIN            */
         END                        /* END OF WHEN STATEMENT        */
      WHEN TOKEN = 'GBPCACHE' THEN DO  /* BEGIN OF WHEN STATEMENT    */
         CALL PARSE_GBPCACHE_PARM   /* CALL CHECKER SUB-ROUTINE     */
         P_GBPCACHE.PARTNO = GBPTKN /* SAVE PARAMETER IN ARRAY.     */
         CURPOS = CURPOS + 2        /* INCREMENT CURSOR POINTER BY 2 */
         ITERATE                    /* DO THE LOOP AGAIN            */
         END                        /* END OF WHEN STATEMENT        */
   /*-----------------------------------------------------------------*/
   /* AFTER EACH TOKEN AND/OR KEYWORD, STORE THE FIRST TOKEN THAT IS */
   /* NOT EITHER A CLAUSE, A PART OF A CLAUSE OR A KEYWORD. THE TOKEN*/
   /* IS THEN USED TO MARK IN THE TEXT DDL WHERE TO SPLIT THE LINE.  */
   /* THIS ALLOWS THE EDIT MACRO TO DELETE THE LINES THAT HAVE BEEN  */
   /* PARSED, SO THEY CAN BE REPLACED WITHOUT AFFECTING OTHER PARMS. */
   /*-----------------------------------------------------------------*/
      OTHERWISE DO
         IF ( CURPOS > ( NPRTPOS + 2 ) ) & ( SAVTOKEN = '' ) THEN DO
            PREV_TOKEN = WORD(DDLSTR,(CURPOS-1))
            SAVTOKEN = TOKEN              /* SET THE SAVTOKEN VAKUE */
            IF ( SAVTOKEN \= ')' ) &,     /* CHECK FOR VALID TOKEN  */
               ( SUBSTR(SAVTOKEN,1,1) \= ')' ) THEN DO
               LM = 'DDL ERROR # 138: INVALID CLAUSE, OR CLAUSE OUT',
                    'OF PLACE'
               SM = 'DDL ERROR # 138'
               CALL DDLERROR YVAL, CL1, SAVTOKEN
               END                        /* END SAVTOKEN \= ')'    */
            'ISREDIT (SPLITRW2,SPLITCL2) = CURSOR' /* SAVE LOCATION  */
            CURPOS = DDLWORDS +1          /* FORCE LEAVING LOOP     */
            END                           /* END IF CURPOS > ....   */
         CURPOS = CURPOS + 1              /* INCREASE CURSOR POSITI */
         ITERATE                          /* LEAVE TOKEN PART LOOP  */
         END                              /* END OF OTHERWISE       */
      END                                 /* END OF SELECT STMT     */
   END                                    /* END OF DO WHILE CURPOS */
RETURN              /* END OF SUBROUTINE PARSE_TABLESPACE_PARTS    */

/*********************************************************************/
/*   SUB-ROUTINE : PARSE_INDEX_PARTS.       NO ARGUMENTS ARE PASSED. */
/*   THIS SUB-ROUTINE WILL PARSE THE DDL FOR ALL NON-SPACE RELATED   */
/*   KEYWORDS.  ONCE A KEYWORD IS FOUND, A SUB-ROUTINE IS CALLED TO  */
/*   VALIDATE THE SPECIFIED CLAUSES. CLAUSES ARE STORED IN ARRAYS.   */
/*********************************************************************/
PARSE_INDEX_PARTS: PROCEDURE EXPOSE   CL1  CC1   CL2   CC2,
          DB2VER       DDLSTR       DDLWORDS    NPRTPOS,
```

```
          CLUSPOS      STOPFLAG     SAVTOKEN    SAVTOKENPOS,
          SPLITRW1     SPLITCL1     SPLITRW2    SPLITCL2,
          YVAL         HIGHEST_PARTNO PARTNO    SC_HEX_VAL,
          DFTSSTOR     DFTSNAME     DFIXSTOR    DFIXNAME    ,
          P_STOR_TYPE. P_STOR_NAME. P_PRIQTY.   P_SECQTY.   ,
          P_N_PRIQTY.  P_N_SECQTY.  P_SEGSIZE   P_N_SEGSIZE ,
          P_ERASE.     P_FREEPAGE.  P_PCTFREE.  P_GBPCACHE. ,
          P_COMPRESS.  P_VALUES.    CNTGBPCACHE CNTVALUES   ,
          CNTPART      CNTUSING     CNTPRIQTY   CNTSECQTY   ,
          CNTCOMPRESS  CNTERASE     CNTFREEPAGE CNTPCTFREE
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE PARSE_INDEX_PARTS BEGINS HERE.                       */
/*-------------------------------------------------------------------*/
/*  SET VARIABLE TO KEEP TRACK OF LAST LEFT OR RIGHT PARENTHESES IN  */
/*  THE DDL.  INITIAL VALUE IS = TO POSITION OF FIRST PART CLAUSE.   */
/*-------------------------------------------------------------------*/
TEMPPOS  = POS('CLUSTER',DDLSTR)          /* FIND CLUSTER  CLAUSE    */
PARENPOS = POS('PART',DDLSTR,TEMPPOS)     /* FIND PART AFTER CLUSTER */
IF ( PARENPOS = Ø ) THEN DO
   LM = 'DDL ERROR # 139: MACRO COULD NOT DETERMINE POSITION OF PART'
   SM = 'DDL ERROR # 139'
   CALL DDLERROR STOPFLAG, CL1, 'CLUSTER', '', 1
   END

/*-------------------------------------------------------------------*/
/*  CHECK THAT THE CLUSTER CLAUSE IS FOLLOWED BY A LEFT PARENTHESES  */
/*  GET AND SAVE CURSOR POSITION FOR THAT LEFT PARENTHESES.          */
/*-------------------------------------------------------------------*/
BEGSPLIT = WORD(DDLSTR,(CLUSPOS+1))           /* GET WORD TO SEARCH  */
IF ( BEGSPLIT \= '(' ) THEN DO                /* CHECKING FOR '('    */
   LM = 'DDL ERROR # 14Ø: EXPECTING TO FIND A LEFT PARENTHESES'
   SM = 'DDL ERROR # 14Ø'
   CALL DDLERROR YVAL, CL1, 'CLUSTER', BEGSPLIT, 1
   END
'ISREDIT CURSOR = ' CL1 Ø                     /* REPOSITION CURSOR   */
'ISREDIT FIND CLUSTER WORD NEXT'              /* POSITION AT CLUSTER */
'ISREDIT FIND "(" WORD NEXT'                  /* POSITION AT LPAREN  */
'ISREDIT (SPLITRW1,SPLITCL1) = CURSOR'        /* SAVE SPLIT LOCATION */

/*-------------------------------------------------------------------*/
/*  PARAGRAPH : LOOP THRU THE DDL LOOKING FOR KEYWORDS.              */
/*  LOOP THRU THE DDL LOOKING FOR CLAUSES AND PARAMETERS. STORE THEM */
/*  ALWAYS ISSUE A FIND FOR THE CURRENT TOKEN TO KEEP THE CURSOR IN  */
/*  SYNC WITH THE PROCESSING.                                        */
/*-------------------------------------------------------------------*/
CURPOS   = CLUSPOS + 2                        /* SET START POSITION  */
SAVTOKEN = ''                                 /* INITIALIZE VAR      */
CNT_COMMA = Ø                                 /* SET COMMA COUNTER=Ø */
DO WHILE ( CURPOS <= DDLWORDS )               /* LOOP THRU THE DDL   */
   TOKEN  = WORD(DDLSTR,CURPOS)
```

```
IF ( TOKEN = ';' ) THEN ,
     'ISREDIT FIND X"'SC_HEX_VAL'" NEXT'
ELSE 'ISREDIT FIND "'TOKEN'" NEXT WORD'
SELECT
   WHEN TOKEN = ',' THEN DO      /* BEGIN OF WHEN STATEMENT       */
      CNT_COMMA = CNT_COMMA + 1           /* INCREASE COMMA CNT  */
      CURPOS = CURPOS + 1        /* INCREMENT CURSOR POINTER BY 2 */
      IF CNT_COMMA > HIGHEST_PARTNO THEN DO
         LM = 'DDL ERROR # 141: COMMA FOUND BEFORE PART CLAUSE'
         SM = 'DDL ERROR # 141'
         CALL DDLERROR YVAL, CL1, ',', '', CNT_COMMA
         END
      ITERATE                    /* DO THE LOOP AGAIN             */
      END                        /* END OF WHEN STATEMENT         */
   WHEN TOKEN = 'PART'  THEN DO  /* BEGIN OF WHEN STATEMENT       */
      PARTNO = WORD(DDLSTR,(CURPOS+1))     /* GET IDX PART NUMBER */
      IF ( DATATYPE(PARTNO,'NUM') \= 1 ) THEN DO
         LM = 'DDL ERROR # 142: COULD NOT GET THE PART NUMBER'
         SM = 'DDL ERROR # 142'
         CALL DDLERROR YVAL, CL1, 'PART', PARTNO
         END
      CURPOS = CURPOS + 2        /* INCREMENT CURSOR POINTER BY 2 */
      IF PARTNO > HIGHEST_PARTNO THEN ,
         HIGHEST_PARTNO = PARTNO        /* SET HIGHEST PART NO    */
      ITERATE                    /* DO THE LOOP AGAIN             */
      END                        /* END OF WHEN STATEMENT         */
   WHEN TOKEN = 'VALUES' THEN DO /* BEGIN OF WHEN STATEMENT       */
      CNTVALUES = CNTVALUES + 1  /* INCREMENT OCCURRENCE COUNTER  */
      N_LPAREN = WORDPOS('(',DDLSTR,CURPOS) /* GET POS NEXT '('   */
      N_RPAREN = WORDPOS(')',DDLSTR,CURPOS) /* GET POS NEXT ')'   */
      NWORDS  = N_RPAREN - N_LPAREN + 1     /* COUNT THE # WORDS  */
      VALUES  = SUBWORD(DDLSTR,N_LPAREN,NWORDS)     /* SET VALUES */
      CALL CHECK_VALUES_PARM               /* CHECK VALUES PARM  */
      P_VALUES.PARTNO = VALUES             /* SAVE VALUES VARS.  */
      CURPOS = CURPOS + NWORDS +1          /* INCREMENT WORD PTR */
      PARENPOS = NEXT_RPAREN               /* SAVE NEW PAREN VAL */
      ITERATE                    /* DO THE LOOP AGAIN             */
      END                        /* END OF WHEN STATEMENT         */
   WHEN TOKEN = 'USING' THEN DO  /* BEGIN OF WHEN STATEMENT       */
      CALL PARSE_USING_PARM      /* CALL CHECKER SUB-ROUTINE      */
      P_STOR_TYPE.PARTNO = U_STOR /* STORE USING KEYWORD IN ARRAY */
      P_STOR_NAME.PARTNO = U_NAME /* STORE USING VALUE  IN ARRAY  */
      CURPOS = CURPOS + 3        /* INCREMENT TOKEN POSITION BY 3 */
      ITERATE                    /* DO THE LOOP AGAIN             */
      END                        /* END OF WHEN STATEMENT         */
   WHEN TOKEN = 'PRIQTY' THEN DO /* BEGIN OF WHEN STATEMENT       */
      CALL PARSE_PRIQTY_PARM     /* CALL CHECKER SUB-ROUTINE      */
      P_PRIQTY.PARTNO   = WORD(DDLSTR,(CURPOS+1)) /* SAVE VALUE   */
      CURPOS = CURPOS + 2        /* INCREMENT TOKEN POSITION BY 2 */
      ITERATE                    /* DO THE LOOP AGAIN             */
```

```
   END                          /* END OF WHEN STATEMENT         */
WHEN TOKEN = 'SECQTY' THEN DO /* BEGIN OF WHEN STATEMENT       */
   CALL PARSE_SECQTY_PARM     /* CALL CHECKER SUB-ROUTINE      */
   P_SECQTY.PARTNO   = WORD(DDLSTR,(CURPOS+1)) /* SAVE VALUE   */
   CURPOS = CURPOS + 2        /* INCREMENT TOKEN POSITION BY 2 */
   ITERATE                    /* DO THE LOOP AGAIN             */
   END                        /* END OF WHEN STATEMENT         */
WHEN TOKEN = 'ERASE' THEN DO  /* BEGIN OF WHEN STATEMENT       */
   CALL PARSE_ERASE_PARM      /* CALL CHECKER SUB-ROUTINE      */
   P_ERASE.PARTNO    = ERASE  /* SAVE THE VALUE                */
   CURPOS = CURPOS + 2        /* INCREMENT TOKEN POSITION BY 2 */
   ITERATE                    /* DO THE LOOP AGAIN             */
   END                        /* END OF WHEN STATEMENT         */
WHEN TOKEN = 'FREEPAGE' THEN DO /* BEGIN OF WHEN STATEMENT     */
   CALL PARSE_FREEPAGE_PARM   /* CALL CHECKER SUB-ROUTINE      */
   P_FREEPAGE.PARTNO = FPG    /* SAVE THE VALUE                */
   CURPOS = CURPOS + 2        /* INCREMENT TOKEN POSITION BY 2 */
   ITERATE                    /* DO THE LOOP AGAIN             */
   END                        /* END OF WHEN STATEMENT         */
WHEN TOKEN = 'PCTFREE' THEN DO /* BEGIN OF WHEN STATEMENT      */
   CALL PARSE_PCTFREE_PARM    /* CALL CHECKER SUB-ROUTINE      */
   P_PCTFREE.PARTNO  = PCT    /* SAVE THE VALUE                */
   CURPOS = CURPOS + 2        /* INCREMENT TOKEN POSITION BY 2 */
   ITERATE                    /* DO THE LOOP AGAIN             */
   END                        /* END OF WHEN STATEMENT         */
WHEN TOKEN = 'GBPCACHE' THEN DO /* BEGIN OF WHEN STATEMENT     */
   CALL PARSE_GBPCACHE_PARM   /* CALL CHECKER SUB-ROUTINE      */
   P_GBPCACHE.PARTNO = GBPTKN /* SAVE THE VALUE                */
   CURPOS = CURPOS + 2        /* INCREMENT TOKEN POSITION BY 2 */
   ITERATE                    /* DO THE LOOP AGAIN             */
   END                        /* END OF WHEN STATEMENT         */
/*-------------------------------------------------------------*/
/* ONLY STORE THE FIRST TOKEN THAT IS NOT EITHER A CLAUSE OR   */
/* A PARAMETER.  THIS TOKEN WILL GIVE THE LOCATION WHERE TO    */
/* PERFORM AN EDIT TEXT SPLIT OPERATION, SO THAT LINES CAN BE  */
/* DELETED AND INSERTED WITHOUT IMPACTING THE REST OF THE DDL. */
/*-------------------------------------------------------------*/
OTHERWISE DO
   IF ( CURPOS > ( CLUSPOS + 2 ) ) & ( SAVTOKEN = '' ) THEN DO
      PREV_TOKEN = WORD(DDLSTR,(CURPOS-1))     /* SET PREVTKN  */
      SAVTOKEN = TOKEN                         /* SET TOKEN    */
      SAVTOKENPOS = CURPOS                     /* SAVE POSTION */
      IF ( SAVTOKEN \= ')' ) THEN DO /* CHECK FOR VALID TOKEN  */
         LM = 'DDL ERROR # 143: EXPECTING A RIGHT PARENTHESES.',
              'CHECK DDL SYNTAX'
         SM = 'DDL ERROR # 143'
         CALL DDLERROR YVAL, CL1, PREV_TOKEN, SAVTOKEN
         END                        /* END SAVTOKEN \= ')'    */
      'ISREDIT (SPLITRW2,SPLITCL2) = CURSOR'   /* SAVE LOCATION*/
      CURPOS = DDLWORDS + 1          /* FORCE LEAVING LOOP     */
```

```
            END                                 /* END OF IF CP > ....    */
         CURPOS = CURPOS + 1                    /* INCREASE CURPOS BY 1   */
         ITERATE                                /* LEAVE TOKEN PART LOOP  */
         END                                    /* END OF OTHERWISE       */
      END                                       /* END OF SELECT STMT     */
   END                                          /* END OF DO WHILE CURPOS */
RETURN                      /* END OF SUB-ROUTINE PARSE_INDEX_PARTS  */

/*********************************************************************/
/*  SUB-ROUTINE CHECK_VALUES_PARM:                                   */
/*  NO ARGUMENTS ARE SPECIFIED.  ALL NEEDED VALUES ARE EXPOSED.      */
/*********************************************************************/
CHECK_VALUES_PARM: PROCEDURE EXPOSE CL1 DDLSTR STOPFLAG YVAL VALUES
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE CHECK_VALUES_PARM BEGINS HERE.  THE ONLY PURPOSE OF  */
/*  THIS ROUTINE IS TO ENSURE THAT THE VALUES PARM CAN BE SPLIT IN   */
/*  CHUNKS OF 70 BYTES, AS DELIMITED BY THE COMMAS IN THE PARM.      */
/*-------------------------------------------------------------------*/
TLX = VALUES                                        /* COPY VALUES  */
LEN_TLX = LENGTH(VALUES)                            /* GET LENGTH   */
IF ( LEN_TLX > 70 ) THEN DO                         /* IF LINE>70   */
   NC = POS(',',TLX)                                /* LOCATE ','   */
   DO WHILE ( NC > 0 )                              /* WHILE FOUND  */
      LEFT_TLX = LEFT(TLX,NC)                       /* GET LEFT STR */
      TLX      = RIGHT(TLX,(LEN_TLX-NC))            /* GET RIGHT STR*/
      LEN_TLX  = LENGTH(TLX)                        /* GET NEW LENG */
      IF ( LENGTH(LEFT_TLX) > 70 ) THEN DO          /* DOUBLE CHECK */
         LM = 'EDIT MACRO ERROR: COULD NOT SPLIT THE VALUES PARM'
         SM = 'MACRO ERROR'
         CALL DDLERROR STOPFLAG, CL1, 'CREATE', 'VALUES'
         END
      NC = POS(',',TLX)                             /* LOCATE NEXT  */
      END                                           /* END DO WHILE */
   END                                           /* END LEN_TLX > 70 */
RETURN                       /* END OF SUB-ROUTINE CHECK_VALUES_PARM */

/*********************************************************************/
/*  SUB-ROUTINE PARSE_USING_PARM :                                   */
/*  NO ARGUMENTS ARE SPECIFIED.  ALL NEEDED VALUES ARE EXPOSED.      */
/*  CALLED BY ROUTINES PARSE_TABLESPACE_PARTS AND PARSE_INDEX_PARTS. */
/*  ALSO CALLED FROM MAIN ROUTINE, PARAGRAPH PARSE_USING CLAUSE FOR. */
/*  NON PARTITIONED TABLESPACES TO SET THE U_STOR AND U_NAME VALUES. */
/*********************************************************************/
PARSE_USING_PARM: PROCEDURE EXPOSE CL1 DDLSTR CURPOS,
       CNTUSING U_STOR U_NAME STOPFLAG YVAL
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE PARSE_USING_PARM BEGINS HERE                         */
/*-------------------------------------------------------------------*/
CNTUSING = CNTUSING + 1
U_STOR = WORD(DDLSTR,(CURPOS+1))
```

```
U_NAME = WORD(DDLSTR,(CURPOS+2))
IF \ ( U_STOR = 'STOGROUP' | U_STOR = 'VCAT') THEN DO      /* CHECK  */
   LM = 'DDL ERROR # 144: WRONG KEYWORD FOR USING CLAUSE'
   SM = 'DDL ERROR # 144'
   CALL DDLERROR YVAL, START_LINE, 'USING', STORTYPE
   END
RETURN                        /* END OF SUB-ROUTINE PARSE_USING_PARM */

/*********************************************************************/
/*  SUB-ROUTINE PARSE_PRIQTY_PARM:                                   */
/*  NO ARGUMENTS ARE SPECIFIED.  ALL NEEDED VALUES ARE EXPOSED.      */
/*********************************************************************/
PARSE_PRIQTY_PARM: PROCEDURE EXPOSE CL1 DDLSTR CURPOS,
        CNTPRIQTY QTY STOPFLAG YVAL
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE PARSE_PRIQTY_PARM BEGINS HERE                        */
/*-------------------------------------------------------------------*/
CNTPRIQTY = CNTPRIQTY + 1
QTY = WORD(DDLSTR,(CURPOS+1))
IF ( DATATYPE(QTY,'NUM') \= 1 ) THEN DO
   LM = 'DDL ERROR # 145: PRIQTY VALUE NOT NUMERIC'
   SM = 'DDL ERROR # 145'
   CALL DDLERROR YVAL, CL1,'PRIQTY', QTY, CNTPRIQTY
   END
RETURN                       /* END OF SUB-ROUTINE PARSE_PRIQTY_PARM */

/*********************************************************************/
/*  SUB-ROUTINE PARSE_SECQTY_PARM:                                   */
/*  NO ARGUMENTS ARE SPECIFIED.  ALL NEEDED VALUES ARE EXPOSED.      */
/*********************************************************************/
PARSE_SECQTY_PARM: PROCEDURE EXPOSE CL1 DDLSTR CURPOS,
        CNTSECQTY QTY STOPFLAG YVAL
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE PARSE_SECQTY_PARM BEGINS HERE.                       */
/*-------------------------------------------------------------------*/
CNTSECQTY = CNTSECQTY + 1
QTY = WORD(DDLSTR,(CURPOS+1))
IF ( DATATYPE(QTY,'NUM') \= 1 ) THEN DO
   LM = 'DDL ERROR # 146: SECQTY VALUE NOT NUMERIC'
   SM = 'DDL ERROR # 146'
   CALL DDLERROR YVAL, CL1, 'SECQTY', QTY, CNTSECQTY
   END
RETURN                       /* END OF SUB-ROUTINE PARSE_SECQTY_PARM */

/*********************************************************************/
/*  SUB-ROUTINE PARSE_COMPRESS_PARM:                                 */
/*  NO ARGUMENTS ARE SPECIFIED.  ALL NEEDED VALUES ARE EXPOSED.      */
/*********************************************************************/
PARSE_COMPRESS_PARM : PROCEDURE EXPOSE CL1 DDLSTR CURPOS,
        CNTCOMPRESS CMPYN STOPFLAG YVAL
```

```
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE PARSE_COMPRESS_PARM BEGINS HERE.                     */
/*-------------------------------------------------------------------*/
CNTCOMPRESS = CNTCOMPRESS + 1
CMPYN = WORD(DDLSTR,(CURPOS+1))
IF \ ( CMPYN = 'YES' | CMPYN = 'NO') THEN DO
   LM = 'DDL ERROR # 147: WRONG KEYWORD FOR COMPRESS'
   SM = 'DDL ERROR # 147'
   CALL DDLERROR STOPFLAG, CL1,'COMPRESS', CMPYN, CNTCOMPRESS
   END
RETURN                      /* END OF SUB-ROUTINE PARSE_COMPRESS_PARM*/

/*********************************************************************/
/*  SUB-ROUTINE PARSE_ERASE_PARM:                                    */
/*  NO ARGUMENTS ARE SPECIFIED.  ALL NEEDED VALUES ARE EXPOSED.      */
/*********************************************************************/
PARSE_ERASE_PARM : PROCEDURE EXPOSE CL1 DDLSTR CURPOS,
        CNTERASE ERASE STOPFLAG YVAL
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE PARSE_ERASE_PARM BEGINS HERE.                        */
/*-------------------------------------------------------------------*/
CNTERASE = CNTERASE + 1
ERASE = WORD(DDLSTR,(CURPOS+1))
IF \( ERASE = 'YES' | ERASE = 'NO') THEN DO
   LM = 'DDL ERROR # 148: WRONG KEYWORD FOR ERASE'
   SM = 'DDL ERROR # 148'
   CALL DDLERROR STOPFLAG, CL1,'ERASE', ERASE, CNTERASE
   END
RETURN                        /* END OF SUB-ROUTINE PARSE_ERASE_PARM */

/*********************************************************************/
/*  SUB-ROUTINE PARSE_FREEPAGE_PARM:                                 */
/*  NO ARGUMENTS ARE SPECIFIED.  ALL NEEDED VALUES ARE EXPOSED.      */
/*********************************************************************/
PARSE_FREEPAGE_PARM: PROCEDURE EXPOSE CL1 DDLSTR CURPOS,
        CNTFREEPAGE FPG STOPFLAG YVAL
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE PARSE_FREEPAGE_PARM BEGINS HERE                      */
/*-------------------------------------------------------------------*/
CNTFREEPAGE = CNTFREEPAGE + 1
FPG = WORD(DDLSTR,(CURPOS+1))
IF ( DATATYPE(FPG,'NUM') \= 1 ) THEN DO
   LM = 'DDL ERROR # 149: FREEPAGE VALUE NOT NUMERIC'
   SM = 'DDL ERROR # 149'
   CALL DDLERROR YVAL, CL1,'FREEPAGE', FPG, CNTFREEPAGE
   END
IF ( FPG < Ø | FPG > 255 ) THEN DO
   LM = 'DDL ERROR # 15Ø: FREEPAGE VALUE OUT OF RANGE(Ø-255)'
   SM = 'DDL ERROR # 15Ø'
   CALL DDLERROR STOPFLAG, CL1,'FREEPAGE', FPG, CNTFREEPAGE
```



```
   END
RETURN                       /* END OF SUB-ROUTINE PARSE_FREEPAGE_PARM*/

/*********************************************************************/
/*  SUB-ROUTINE PARSE_PCTFREE_PARM:                                  */
/*  NO ARGUMENTS ARE SPECIFIED.  ALL NEEDED VALUES ARE EXPOSED.      */
/*********************************************************************/
PARSE_PCTFREE_PARM: PROCEDURE EXPOSE CL1 DDLSTR CURPOS,
        CNTPCTFREE PCT STOPFLAG YVAL
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE PARSE_PCTFREE_PARM BEGINS HERE                       */
/*-------------------------------------------------------------------*/
CNTPCTFREE = CNTPCTFREE + 1
PCT = WORD(DDLSTR,(CURPOS+1))
IF ( DATATYPE(PCT,'NUM') \= 1 ) THEN DO
   LM = 'DDL ERROR # 151: PCTFREE VALUE NOT NUMERIC'
   SM = 'DDL ERROR # 151'
   CALL DDLERROR YVAL, CL1,'PCTFREE', PCT, CNTPCTFREE
   END
IF ( PCT < Ø | PCT > 99 ) THEN DO
   LM = 'DDL ERROR # 152: PCTFREE VALUE OUT OF RANGE(Ø-99)'
   SM = 'DDL ERROR # 152'
   CALL DDLERROR STOPFLAG, CL1,'PCTFREE', PCT, CNTPCTFREE
   END
RETURN                       /* END OF SUB-ROUTINE PARSE_PCTFREE_PARM */

/*********************************************************************/
/*  SUB-ROUTINE PARSE_GBPCHACHE_PARM:                                */
/*  NO ARGUMENTS ARE SPECIFIED.  ALL NEEDED VALUES ARE EXPOSED.      */
/*********************************************************************/
PARSE_GBPCACHE_PARM : PROCEDURE EXPOSE CL1 DDLSTR CURPOS,
        CNTGBPCACHE GBPTKN STOPFLAG YVAL
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE PARSE_GBPCACHE_PARM BEGINS HERE                      */
/*-------------------------------------------------------------------*/
CNTGBPCACHE = CNTGBPCACHE + 1
GBPTKN = WORD(DDLSTR,(CURPOS+1))
IF \ ( GBPTKN = 'CHANGED' | GBPTKN = 'ALL') THEN DO
   LM = 'DDL ERROR # 153: WRONG KEYWORD FOR GBPCACHE'
   SM = 'DDL ERROR # 153'
   CALL DDLERROR STOPFLAG, CL1,'GBPCACHE', GBPTKN, CNTGBPCACHE
   END
RETURN                       /* END OF SUB-ROUTINE PARSE_GBPCACHE_PARM*/

/*********************************************************************/
/*  SUB-ROUTINE UPDATE_TABLESPACE_PARTS_DDL:  NO PARAMETERS PASSED.  */
/*********************************************************************/
UPDATE_TABLESPACE_PARTS_DDL: PROCEDURE EXPOSE CL1 DB2VER DDLSTR,
          BEGSPLIT     SAVTOKEN     NUMPARTS     PARTFLAG   YVAL,
          DFTSSTOR     DFTSNAME     DFIXSTOR     DFIXNAME  SETDFLAG,
```

```
          SPLITRW1     SPLITCL1     SPLITRW2     SPLITCL2,
          P_STOR_TYPE. P_STOR_NAME. P_PRIQTY.    P_SECQTY. ,
          P_N_PRIQTY.  P_N_SECQTY.  P_ERASE.     P_FREEPAGE. ,
          P_PCTFREE.   P_GBPCACHE.  P_COMPRESS.  P_VALUES.
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE UPDATE_TABLESPACE_PARTS_DDL BEGINS HERE              */
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE LOGIG:  DELETE OLD DDL CONTAINING PARTS INFORMATION. */
/*  USE VARIABLES SPLITRW1, SPLITCL1, SPLITRW2, SPLITCL2 SET BY THE  */
/*  PARSE_TABLESPACE_PARTS SUBROUTINE.  INSERT THE NEW LINES.        */
/*-------------------------------------------------------------------*/
'ISREDIT TSPLIT' SPLITRW2 SPLITCL2-1   /* SPLIT BOTTOM LINE FIRST    */
'ISREDIT CURSOR = ' SPLITRW2 1         /* REPOSITION THE CURSOR      */
'ISREDIT (BOTTOMLN) = LINENUM ' .ZCSR  /* SET BOTTOM LABEL FOR LINE  */
'ISREDIT TSPLIT' SPLITRW1 SPLITCL1+1    /* SPLIT TOP LINE SECOND     */
'ISREDIT CURSOR = ' SPLITRW1+1 1        /* REPOSITION THE CURSOR     */
'ISREDIT (TOPLN) = LINENUM ' .ZCSR      /* SET TOP LABEL FOR LINE    */
'ISREDIT DELETE ' TOPLN BOTTOMLN        /* DELETE LINES USING LABELS */
TOPLN = TOPLN - 1                       /* SUBTRACT 1 FROM TOP LINE  */
DO CNT=1 TO NUMPARTS BY 1              /* LOOP THRU ALL THE PARTS    */
   TL = Ø                              /* COUNTER FOR OPTIONAL LINES */
   /* INSERT REPLACEMENT LINE WITH PART CLAUSE.                     */
   IF ( CNT = 1) THEN TLX = '  PART ' CNT
   ELSE               TLX = ', PART ' CNT
   CALL INSERT_LINE 'AFTER', TOPLN, TLX
   /* INSERT REPLACEMENT LINE WITH USING CLAUSE.  CHECK FOR DEFAULTS */
   TLX = 'USING  ' P_STOR_TYPE.CNT P_STOR_NAME.CNT
   CALL INSERT_LINE 'AFTER', TOPLN+1, TLX
   U_STOR = P_STOR_TYPE.CNT
   U_NAME = P_STOR_NAME.CNT
   CALL CHECK_USING_PARM TOPLN, 'TS',U_STOR, U_NAME, 1
   /* INSERT REPLACEMENT LINE WITH PRIQTY CLAUSE.                   */
   TLX = 'PRIQTY ' P_N_PRIQTY.CNT
   CALL INSERT_LINE 'AFTER', TOPLN+2, TLX
   IF ( P_PRIQTY.CNT \= P_N_PRIQTY.CNT ) THEN DO
      TLX = 'PRIQTY CHANGED FROM' P_PRIQTY.CNT 'TO' P_N_PRIQTY.CNT
      CALL WRITE_MSG TLX, TOPLN+3
      END
   /* IF PRIQTY LESS THAN SECQTY, WRITE A NOTE                      */
   IF ( P_N_PRIQTY.CNT <  P_N_SECQTY.CNT ) THEN DO
      TLX = 'PRIQTY LESS THAN SECQTY.  VERIFY'
      CALL WRITE_NOTE TLX, TOPLN+3
      END
   /* INSERT REPLACEMENT LINE WITH SECQTY CLAUSE.                   */
   TLX = 'SECQTY ' P_N_SECQTY.CNT
   CALL INSERT_LINE 'AFTER', TOPLN+3, TLX
   IF ( P_SECQTY.CNT \= P_N_SECQTY.CNT ) THEN DO
      TLX = 'SECQTY CHANGED FROM' P_SECQTY.CNT 'TO' P_N_SECQTY.CNT
      CALL WRITE_MSG TLX, TOPLN+4
      END
```

```
   /* INSERT REPLACEMENT LINE WITH ERASE CLAUSE                      */
   IF ( P_ERASE.CNT \= '' ) THEN DO
      TLX = 'ERASE ' P_ERASE.CNT
      CALL INSERT_LINE 'AFTER', TOPLN+4, TLX
      TL = TL + 1               /* INCREASE OPTIONAL LINE CNT       */
      END
   /* INSERT REPLACEMENT LINE WITH FREEPAGE CLAUSE                   */
   IF ( P_FREEPAGE.CNT \= '' ) THEN DO
      TLX = 'FREEPAGE ' P_FREEPAGE.CNT
      CALL INSERT_LINE 'AFTER', TOPLN+4+TL, TLX
      TL = TL + 1               /* INCREASE OPTIONAL LINE CNT       */
      END
   /* INSERT REPLACEMENT LINE WITH PCTFREE CLAUSE                    */
   IF ( P_PCTFREE.CNT \= '' ) THEN DO
      TLX = 'PCTFREE ' P_PCTFREE.CNT
      CALL INSERT_LINE 'AFTER', TOPLN+4+TL, TLX
      TL = TL + 1               /* INCREASE OPTIONAL LINE CNT       */
      END
   /* INSERT REPLACEMENT LINE WITH GBPCACHE CLAUSE.                  */
   IF ( P_GBPCACHE.CNT \= '' ) THEN DO
      TLX = 'GBPCACHE ' P_GBPCACHE.CNT
      CALL INSERT_LINE 'AFTER', TOPLN+4+TL, TLX
      TL = TL + 1               /* INCREASE OPTIONAL LINE CNT       */
      END
   /* INSERT REPLACEMENT LINE WITH COMPRESS CLAUSE.                  */
   IF ( P_COMPRESS.CNT \= '' ) THEN DO
      TLX = 'COMPRESS ' P_COMPRESS.CNT
      CALL INSERT_LINE 'AFTER', TOPLN+4+TL, TLX
      TL = TL + 1               /* INCREASE OPTIONAL LINE CNT       */
      END
   TOPLN = TOPLN + TL + 4                 /* SET TOP OF NEXT PART   */
   END                                    /* END OF DO WHILE CNT    */
RETURN             /* END OF SUB-ROUTINE UPDATE_TABLESPACE_PARTS_DDL*/

/*********************************************************************/
/*  SUB-ROUTINE UPDATE_TABLESPACE_DDL:  A PARAMETER IS PASSED TO     */
/*  INDICATE IF THE ROUTINE IS CALLED FOR A CREATE OR ALTER STMT.    */
/*********************************************************************/
UPDATE_TABLESPACE_DDL: PROCEDURE EXPOSE CL1 DB2VER DDLSTR,
          PRIQTYPOS    SECQTYPOS    SEGSZ,
          BEGSPLIT     SAVTOKEN     NUMPARTS    PARTFLAG,
          PRIQTY       SECQTY       BLANK       U_STOR U_NAME,
          P_STOR_TYPE. P_STOR_NAME. P_PRIQTY.   P_SECQTY. ,
          P_N_PRIQTY.  P_N_SECQTY.  P_ERASE.    P_FREEPAGE. ,
          P_PCTFREE.   P_GBPCACHE.  P_COMPRESS. P_VALUES. ,
          P_SEGSIZE    P_N_SEGSIZE
ARG DDL_FUNCTION
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE UPDATE_TABLESPACE_DDL (FOR NON-PARTITIONED TSPACES)  */
/*  OR ALTER TABLESPACE STATEMENTS WITH QTY VALUES SPECIFIED.        */
```

```
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE LOGIG: LOOK FOR THE PRIQTY AND SECQTY CLAUSES. FOR   */
/*  EACH QTY CLAUSE, THE EDIT MACRO WILL DO THE FOLLOWING :          */
/*  1)  FIND THE QTY CLAUSE, EITHER PRIQTY OR SECQTY. SAVE THE CURSOR*/
/*      POSITION, AS POS1.                                           */
/*  2)  STORE THE CONTENTS OF THE LINE TO DETERMINE IF THE QTY CLAUSE*/
/*      IS THE FIRST TOKEN IN THE LINE OR NOT.                       */
/*  3)  FIND THE QTY VALUE FOLLOWING THE QTY CLAUSE AND SAVE THE     */
/*      CURSOR POSITION AS POSITION POS2.                            */
/*  4)  IF THERE ARE ADDITIONAL PARAMETERS AFTER POS2, THEN THE EDIT */
/*      MACRO WILL SPLIT THE TEXT FROM POS2 ON INTO A NEW LINE.      */
/*  5)  IF THE QTY CLAUSE IS THE FIRST TOKEN ON THE LINE WHERE IT WAS*/
/*      FOUND, NO LINE SPLIT WILL BE DONE. OTHERWISE, SPLIT THE QTY  */
/*      CLAUSE AND PARAMETER INTO A NEW LINE.                        */
/*  6)  DEPENDING ON HOW THE LINE GOT SPLIT, EITHER REPLACE THE XL1  */
/*      LINE OR THE XL1+1 LINE WITH THE QTY COMPUTED BY THIS REXXEXEC*/
/*  7)  IF THERE WAS A CHANGE IN VALUE, WRITE AN INFORMATIONAL NOTE. */
/*-------------------------------------------------------------------*/

/*-------------------------------------------------------------------*/
/*  IF THE QTY WAS NOT FOUND, AN EXPLICIT VALUE IS INSERTED AFTER QTY*/
/*-------------------------------------------------------------------*/
IF ( PRIQTYPOS > Ø ) THEN DO              /* WAS PRIQTY SPECIFIED ?  */
   IF ( P_PRIQTY.1 \= P_N_PRIQTY.1 ) THEN DO
      'ISREDIT CURSOR = ' CL1 Ø                /* REPOSITION CURSOR. */
      'ISREDIT FIND PRIQTY NEXT WORD'          /* FIND KEYWORD       */
      'ISREDIT (XL1,XC1) = CURSOR'             /* GET LINE NUMBER    */
      'ISREDIT (LNX) = LINE' XL1               /* GET LINE CONTENTS  */
      PARSE VAR LNX TOKEN .                    /* PARSE FIRST TOKEN  */
      'ISREDIT FIND "'PRIQTY'" NEXT WORD'      /* FIND THE QTY VALUE */
      'ISREDIT FIND '''BLANK''' NEXT'          /* FIND NEXT BLANK    */
      'ISREDIT (XL2,XC2) = CURSOR'             /* GET NEW CURSOR POS */
      LNXTAIL = SUBSTR(LNX,XC2)                /* GET TAIL OF LINE   */
      IF ( WORDS(LNXTAIL) \= Ø ) THEN 'ISREDIT TSPLIT' XL2 XC2
      TLX = '     PRIQTY ' P_N_PRIQTY.1
      /* IF FIRST TOKEN NOT 'PRIQTY', SPLIT LINE & REPLACE THE XL1+1 */
      /* LINE WITH THE NEW PRIQTY VALUE. IF FIRST TOKEN IS PRIQTY,   */
      /* THE REPLACE THE LINE.  IF NOT, INSERT AFTER LINE            */
      IF ( TOKEN \= 'PRIQTY' ) THEN DO
         'ISREDIT TSPLIT' XL1 XC1-1            /* SPLIT THE LINE     */
         XL1 = XL1 + 1                         /* RESET LINE POINTER */
         END
      CALL INSERT_LINE 'SAME', XL1, TLX
      /* WRITE NOTE OF CHANGES.                                      */
      TLX = 'PRIQTY CHANGED FROM ' P_PRIQTY.1 'TO' P_N_PRIQTY.1
      CALL WRITE_MSG TLX, XL1
      END                     /* END OF P_PRIQTY.1 \= P_N_PRIQTY.1   */
   END                        /* END OF PRIQTYPOS > Ø PARAGRAPH      */
ELSE DO                       /* PRIQTYPOS = Ø, I.E., NOT SPECIFIED  */
    IF ( DDL_FUNCTION = 'CREATE' ) THEN DO
```

```
      'ISREDIT CURSOR = ' CL1 Ø              /* REPOSITION CURSOR.   */
      'ISREDIT FIND USING NEXT WORD'         /* FIND KEYWORD         */
      'ISREDIT FIND "'U_STOR'" NEXT WORD'
      'ISREDIT FIND "'U_NAME'" NEXT WORD'
      'ISREDIT FIND '''BLANK''' NEXT'        /* FIND NEXT BLANK      */
      'ISREDIT (XL1,XC1) = CURSOR'           /* GET NEW CURSOR POS   */
      'ISREDIT TSPLIT' XL1 XC1
      TLX = 'PRIQTY ' P_N_PRIQTY.1
      CALL INSERT_LINE 'AFTER', XL1, TLX
      /* WRITE NOTE INDICATING PRIQTY WAS INSERTED.                  */
      TLX = 'PRIQTY CLAUSE INSERTED BY EDIT MACRO'
      CALL WRITE_NOTE TLX, XL1+1
      END                               /* END DDL_FUNCTION ='CREATE' */
   END                                  /* END ELSE PRIQTYPOS > Ø     */

/*-------------------------------------------------------------------*/
/*  PARAGRAPH : UPDATE DDL WITH NEW SECQTY VALUE IF CHANGED          */
/*-------------------------------------------------------------------*/
IF ( SECQTYPOS > Ø ) THEN DO              /* WAS SECQTY SPECIFIED ?  */
   IF ( P_SECQTY.1 \= P_N_SECQTY.1 ) THEN DO
      'ISREDIT CURSOR = ' CL1 Ø              /* REPOSITION CURSOR.   */
      'ISREDIT FIND SECQTY NEXT'             /* FIND KEYWORD         */
      'ISREDIT (XL1,XC1) = CURSOR'           /* GET LINE NUMBER      */
      'ISREDIT (LNX) = LINE' XL1             /* GET LINE CONTENTS    */
      PARSE VAR LNX TOKEN .                  /* PARSE FIRST TOKEN    */
      'ISREDIT FIND "'SECQTY'" NEXT WORD'    /* FIND THE QTY VALUE   */
      'ISREDIT FIND '''BLANK''' NEXT'        /* FIND NEXT BLANK      */
      'ISREDIT (XL2,XC2) = CURSOR'           /* GET NEW CURSOR POS   */
      LNXTAIL = SUBSTR(LNX,XC2)              /* GET TAIL OF LINE.    */
      IF ( WORDS(LNXTAIL) \= Ø ) THEN 'ISREDIT TSPLIT' XL2 XC2
      TLX = 'SECQTY ' P_N_SECQTY.1
      /* IF FIRST TOKEN NOT 'SECQTY', SPLIT LINE & REPLACE THE XL1+1 */
      /* LINE WITH THE NEW SECQTY VALUES.                            */
      /* IF FIRST TOKEN IS 'SECQTY', SPLIT LINE AND REPLACE THE LINE */
      IF ( TOKEN \= 'SECQTY' ) THEN DO
         'ISREDIT TSPLIT' XL1 XC1-1            /* SPLIT LINE         */
         XL1 = XL1 + 1                         /* RESET LINE POINTER */
         END
      CALL INSERT_LINE 'SAME', XL1, TLX
      /* WRITE NOTE OF CHANGES.                                      */
      TLX = 'SECQTY CHANGED FROM ' P_SECQTY.1 'TO' P_N_SECQTY.1
      CALL WRITE_MSG TLX, XL1
      END                   /* END OF IF P_SECQTY \= P_N_SECQTY      */
   END                      /* END OF IF SECQTYPOS > Ø               */
ELSE DO             /* NO SECONDARY QTY WAS FOUND, INSERTING LINE    */
    IF ( DDL_FUNCTION = 'CREATE' ) THEN DO
      'ISREDIT CURSOR = ' CL1 Ø              /* REPOSITION CURSOR.   */
      'ISREDIT FIND PRIQTY NEXT WORD'        /* FIND KEYWORD         */
      'ISREDIT (XL1,XC1) = CURSOR'           /* GET NEW CURSOR POS   */
      TLX = 'SECQTY ' P_N_SECQTY.1
```

```
      CALL INSERT_LINE 'AFTER', XL1, TLX
      /* WRITE NOTE INDICATING SECQTY WAS INSERTED.                  */
      TLX = 'SECQTY CLAUSE INSERTED BY EDIT MACRO'
      CALL WRITE_NOTE TLX, XL1+1
      END                           /* END OF DDL_FUNCTION='CREATE' */
   END                              /* END OF ELSE SECQTYPOS > Ø    */

/*-------------------------------------------------------------------*/
/*  PARAGRAPH : IF PRIQTY LESS THAN SECQTY, WRITE A NOTE             */
/*-------------------------------------------------------------------*/
   IF ( P_N_PRIQTY.1 <  P_N_SECQTY.1 ) & ,
      ( DDL_FUNCTION = 'CREATE' )  THEN DO
      'ISREDIT CURSOR = ' CL1 Ø               /* REPOSITION CURSOR. */
      'ISREDIT FIND PRIQTY NEXT WORD'         /* FIND KEYWORD       */
      'ISREDIT (XL1,XC1) = CURSOR'            /* GET LINE NUMBER    */
      TLX = 'PRIQTY LESS THAN SECQTY.  VERIFY'
      CALL WRITE_NOTE TLX, XL1
      END

/*-------------------------------------------------------------------*/
/*  PARAGRAPH : UPDATE DDL WITH NEW SEGSIZE VALUE IF CHANGED         */
/*-------------------------------------------------------------------*/
IF ( P_SEGSIZE \= P_N_SEGSIZE ) & ( DDL_FUNCTION = 'CREATE' )  THEN DO
   'ISREDIT CURSOR = ' CL1 Ø
   'ISREDIT FIND SEGSIZE NEXT'
   'ISREDIT (XL1,XC1) = CURSOR'
   'ISREDIT (LNX) = LINE' XL1
   PARSE VAR LNX TOKEN .
   'ISREDIT FIND "'SEGSZ'" NEXT WORD'
   'ISREDIT FIND '''BLANK''' NEXT'
   'ISREDIT (XL2,XC2) = CURSOR'
   LNXTAIL = SUBSTR(LNX,XC2)
   IF ( WORDS(LNXTAIL) \= Ø ) THEN 'ISREDIT TSPLIT' XL2 XC2
   TLX = '     SEGSIZE ' P_N_SEGSIZE
   IF ( TOKEN \= 'SEGSIZE' ) THEN DO
      'ISREDIT TSPLIT' XL1 XC1-1
      XL1 = XL1 + 1
      END
   CALL INSERT_LINE 'SAME', XL1, TLX
   /* WRITE NOTE OF SEGSIZE CHANGES                                  */
   TLX = 'SEGSIZE VALUE CHANGED FROM ' P_SEGSIZE 'TO' P_N_SEGSIZE
   CALL WRITE_MSG TLX, XL1
   END
RETURN              /* END OF SUB-ROUTINE UPDATE_TABLESPACE_DDL.    */

/*********************************************************************/
/*  SUB-ROUTINE UPDATE_INDEX_PARTS_DDL:     NO PARAMETERS ARE PASSED.*/
/*********************************************************************/
UPDATE_INDEX_PARTS_DDL: PROCEDURE EXPOSE CL1 DDLSTR DB2VER,
          PARTFLAG     YVAL         STOPFLAG    SETDFLAG,
```

```
          DFTSSTOR     DFTSNAME     DFIXSTOR    DFIXNAME,
          SPLITRW1     SPLITCL1     SPLITRW2    SPLITCL2,
          P_STOR_TYPE. P_STOR_NAME. P_PRIQTY.   P_SECQTY. ,
          P_N_PRIQTY.  P_N_SECQTY.  P_SEGSIZE   P_N_SEGSIZE ,
          P_ERASE.     P_FREEPAGE.  P_PCTFREE.  P_GBPCACHE. ,
          P_COMPRESS.  P_VALUES.    NUMPARTS
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE UPDATE_INDEX_PARTS_DDL BEGINS HERE                   */
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE LOGIG:  POSITION THE CURSOR AT THE PLACE IN THE FILE */
/*  WHERE AFTER 'CLUSTER (' IS SPECIFIED.  SPLIT THAT LINE.  DELETE  */
/*  THE LINES CONTAINING OLD PART CLAUSES.  INSERT THE NEW LINES.    */
/*-------------------------------------------------------------------*/
'ISREDIT TSPLIT' SPLITRW2 SPLITCL2-1    /* SPLIT BOTTOM LINE FIRST   */
'ISREDIT CURSOR = ' SPLITRW2 1          /* REPOSITION THE CURSOR     */
'ISREDIT (BOTTOMLN) = LINENUM ' .ZCSR   /* SET BOTTOM LABEL FOR LINE */
'ISREDIT TSPLIT' SPLITRW1 SPLITCL1+1    /* SPLIT TOP LINE SECOND     */
'ISREDIT CURSOR = ' SPLITRW1+1 1        /* REPOSITION THE CURSOR     */
'ISREDIT (TOPLN) = LINENUM ' .ZCSR      /* SET TOP LABEL FOR LINE    */
'ISREDIT DELETE ' TOPLN BOTTOMLN        /* DELETE LINES USING LABELS */
TOPLN = TOPLN - 1                       /* SUBTRACT 1 FROM TOPLN CNT */
DO CNT=1 TO NUMPARTS BY 1              /* LOOP THRU ALL THE PARTS    */
   TL = Ø                              /* COUNTER FOR OPTIONAL LINES */
   /* INSERT REPLACEMENT LINE WITH PART CLAUSE.                      */
   IF ( CNT = 1 ) THEN TLX = '  PART ' CNT          /* SET OUT LINE */
   ELSE                TLX = ', PART ' CNT          /* SET OUT LINE */
   CALL INSERT_LINE 'AFTER', TOPLN, TLX             /* WRITE T LINE */
   /* INSERT REPLACEMENT LINE (OR LINES) WITH VALUE CLAUSES. CHECK   */
   /* THAT THE VALUES PARAMETER FIT WITHIN A 72 CHAR LINE.           */
   TLX = 'VALUES ' P_VALUES.CNT                     /* SET OUT LINE */
   LEN_TLX = LENGTH(TLX)                            /* GET LENGTH   */
   NVL = 1                                          /* #VALUE LINES */
   IF ( LEN_TLX > 7Ø ) THEN DO                      /* IF LINE>7Ø   */
      NVL = Ø                                       /* SET IT TO Ø  */
      NC = POS(',',TLX)                             /* LOCATE ','   */
      DO WHILE ( NC > Ø )                           /* WHILE FOUND  */
         LEFT_TLX = LEFT(TLX,NC)                    /* GET LEFT STR */
         TLX      = RIGHT(TLX,(LEN_TLX-NC))         /* GET RIGHT STR*/
         LEN_TLX  = LENGTH(TLX)                     /* GET NEW LENG */
         IF ( LENGTH(LEFT_TLX) > 7Ø ) THEN DO       /* DOUBLE CHECK */
            LM = 'EDIT MACRO ERROR: COULD NOT SPLIT THE VALUES PARM'
            SM = 'MACRO ERROR'
            CALL DDLERROR STOPFLAG, CL1, 'CREATE', 'INDEX'
            END
         NVL = NVL + 1                              /* ADD 1 TO NVL */
         CALL INSERT_LINE 'AFTER', TOPLN+NVL,LEFT_TLX /* WRT LEFT STR */
         NC = POS(',',TLX)                          /* LOCATE NEXT  */
         END                                        /* END DO WHILE */
      NVL = NVL + 1                                 /* ADD 1 TO NVL */
      CALL INSERT_LINE 'AFTER', TOPLN+NVL, TLX      /* WRT RIGHT ST */
```

```
   END                                        /* END LEN_TLX > 7Ø */
ELSE CALL INSERT_LINE 'AFTER', TOPLN+NVL, TLX /* WRITE THE VALUES */
/* INSERT REPLACEMENT LINE WITH USING CLAUSE.                     */
TLX = 'USING ' P_STOR_TYPE.CNT P_STOR_NAME.CNT    /* SET OUT LINE */
CALL INSERT_LINE 'AFTER', TOPLN+NVL+1, TLX        /* WRITE T LINE */
/* CALL ROUTINE TO VALIDATE THE STORAGE TYPE AND NAME. WRITE NOTE */
STORTYPE =  P_STOR_TYPE.CNT                   /* SET WORK VARIABLE */
STORNAME =  P_STOR_NAME.CNT                   /* SET WORK VARIABLE */
CALL CHECK_USING_PARM TOPLN+NVL+1, 'IX', STORTYPE, STORNAME, 1
/* INSERT REPLACEMENT LINE WITH PRIQTY CLAUSE.                    */
TLX = 'PRIQTY ' P_N_PRIQTY.CNT                    /* SET OUT LINE */
CALL INSERT_LINE 'AFTER', TOPLN+NVL+2, TLX        /* WRITE T LINE */
IF ( P_PRIQTY.CNT \= P_N_PRIQTY.CNT ) THEN DO
   TLX = 'PRIQTY CHANGED FROM' P_PRIQTY.CNT 'TO' P_N_PRIQTY.CNT
   CALL WRITE_MSG TLX, TOPLN+NVL+3
   END
/* IF PRIQTY LESS THAN SECQTY, WRITE A NOTE                       */
IF ( P_N_PRIQTY.CNT <  P_N_SECQTY.CNT ) THEN DO
   TLX = 'PRIQTY LESS THAN SECQTY.  VERIFY'
   CALL WRITE_NOTE TLX, TOPLN+NVL+3
   END
/* INSERT REPLACEMENT LINE WITH SECQTY CLAUSE.                    */
TLX = 'SECQTY ' P_N_SECQTY.CNT                    /* SET OUT LINE */
CALL INSERT_LINE 'AFTER', TOPLN+NVL+3, TLX        /* WRITE T LINE */
IF ( P_SECQTY.CNT \= P_N_SECQTY.CNT ) THEN DO
   TLX = 'SECQTY CHANGED FROM' P_SECQTY.CNT 'TO' P_N_SECQTY.CNT
   CALL WRITE_MSG TLX, TOPLN+NVL+4
   END
/* INSERT REPLACEMENT LINE WITH ERASE CLAUSE.                     */
IF ( P_ERASE.CNT \= '' ) THEN DO
   TLX = 'ERASE ' P_ERASE.CNT                   /* SET OUT LINE */
   CALL INSERT_LINE 'AFTER', TOPLN+NVL+4, TLX    /* WRITE T LINE */
   TL = TL + 1                     /* INCREASE OPTIONAL LINE CNT */
   END
/* INSERT REPLACEMENT LINE WITH FREEPAGE CLAUSE.                  */
IF ( P_FREEPAGE.CNT \= '' ) THEN DO
   TLX = 'FREEPAGE ' P_FREEPAGE.CNT              /* SET OUT LINE */
   CALL INSERT_LINE 'AFTER', TOPLN+NVL+4+TL, TLX  /* WRITE T LINE */
   TL = TL + 1                     /* INCREASE OPTIONAL LINE CNT */
   END
/* INSERT REPLACEMENT LINE WITH PCTFREE CLAUSE.                   */
IF ( P_PCTFREE.CNT \= '' ) THEN DO
   TLX = 'PCTFREE ' P_PCTFREE.CNT                /* SET OUT LINE */
   CALL INSERT_LINE 'AFTER', TOPLN+NVL+4+TL, TLX  /* WRITE T LINE */
   TL = TL + 1                     /* INCREASE OPTIONAL LINE CNT */
   END
/* INSERT REPLACEMENT LINE WITH GBPCACHE CLAUSE                   */
IF ( P_GBPCACHE.CNT \= '' ) THEN DO
   TLX = 'GBPCACHE ' P_GBPCACHE.CNT              /* SET OUT LINE */
   CALL INSERT_LINE 'AFTER', TOPLN+NVL+4+TL, TLX  /* WRITE T LINE */
```

```
      TL = TL + 1                       /* INCREASE OPTIONAL LINE CNT */
      END
   TOPLN = TOPLN + NVL + 4 + TL            /* SET TOP OF NEXT PART   */
   END                                     /* END OF DO WHILE CNT    */
RETURN               /* END OF SUB-ROUTINE UPDATE_INDEX_PARTS_DDL    */

/*********************************************************************/
/*  SUB-ROUTINE UPDATE_INDEX_DDL:  A PARAMETER IS PASSED TO INDICATE */
/*  IF THIS ROUTINE IS CALLED FOR AN ALTER OR A CREATE STATEMENT.    */
/*********************************************************************/
UPDATE_INDEX_DDL: PROCEDURE EXPOSE CL1 DB2VER DDLSTR,
          PRIQTYPOS    SECQTYPOS    PRIQTY      SECQTY    BLANK ,
          BEGSPLIT     SAVTOKEN     PARTFLAG,
          P_STOR_TYPE. P_STOR_NAME. P_PRIQTY.   P_SECQTY. ,
          P_N_PRIQTY.  P_N_SECQTY.  P_SEGSIZE   P_N_SEGSIZE ,
          P_ERASE.     P_FREEPAGE.  P_PCTFREE.  P_GBPCACHE. ,
          P_COMPRESS.  P_VALUES.
ARG DDL_FUNCTION
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE UPDATE_INDEX_DDL BEGINS HERE                         */
/*-------------------------------------------------------------------*/
/*  THE LOGIC FOR THIS SECTION OF CODE IS AS FOLLOWS : THE EDIT MACRO*/
/*  KNOWS THAT THE DDL IS FOR A NON-PARTITIONED INDEXSPACE, SINCE    */
/*  THERE IS NO PART CLAUSE IN THE DDL.  THE PRIQTY & SECQTY CLAUSES */
/*  MAY HAVE BEEN SPECIFIED OR NOT, SO THIS SUB-ROUTINE CHECKS FOR   */
/*  EACH ONE OF THEM.   FOR EACH QTY CLAUSE, THIS SUB-ROUTINE WILL : */
/*-------------------------------------------------------------------*/
/*  1) FIND THE QTY CLAUSE, EITHER PRIQTY OR SECQTY. SAVE THE CURSOR */
/*     POSITION IN VARIABLE POS1.                                    */
/*  2) STORE THE CONTENTS OF THE LINE TO DETERMINE IF THE QTY CLAUSE */
/*     IS THE FIRST TOKEN IN THE LINE OR NOT.                        */
/*  3) FIND THE QTY VALUE FOLLOWING THE QTY CLAUSE AND SAVE THE      */
/*     CURSOR POSITION IN VARIABLE POS2.                             */
/*  4) IF THERE ARE ADDITIONAL PARAMETERS AFTER POS2 ON, THEN PROCEED*/
/*     TO SPLIT TEXT LINE FORWARD FROM POS2 INTO A NEW LINE.         */
/*  5) IF THE QTY CLAUSE IS THE FIRST TOKEN ON THE LINE WHERE IT WAS */
/*     FOUND, DON'T DO A TEXT SPLIT FROM POS1                        */
/*  6) DEPENDING ON HOW THE LINE GOT SPLIT, EITHER REPLACE THE XL1   */
/*     LINE OR THE XL1+1 LINE WITH THE INFORMATION GENERATED BY MACRO*/
/*-------------------------------------------------------------------*/

/*-------------------------------------------------------------------*/
/*  PARAGRAPH : UPDATE PRIQTY VALUE IF NECESSARY.                    */
/*-------------------------------------------------------------------*/
IF ( PRIQTYPOS > Ø ) THEN DO
   IF ( P_PRIQTY.1 \= P_N_PRIQTY.1 ) & ,
      ( DDL_FUNCTION = 'CREATE' ) THEN DO
      'ISREDIT CURSOR = ' CL1 Ø
      'ISREDIT FIND PRIQTY NEXT '
      'ISREDIT (XL1,XC1) = CURSOR'
```

```
      'ISREDIT (LNX) = LINE' XL1
      PARSE VAR LNX TOKEN .
      'ISREDIT FIND "'PRIQTY'" NEXT WORD'
      'ISREDIT FIND '''BLANK''' NEXT'
      'ISREDIT (XL2,XC2) = CURSOR'
      LNXTAIL = SUBSTR(LNX,XC2)
      IF ( WORDS(LNXTAIL) \= Ø ) THEN 'ISREDIT TSPLIT' XL2 XC2
      TLX = 'PRIQTY ' P_N_PRIQTY.1
      IF ( TOKEN \= 'PRIQTY' ) THEN DO
         'ISREDIT TSPLIT' XL1 XC1-1
         CALL INSERT_LINE 'SAME', XL1+1, TLX
         END
      ELSE CALL INSERT_LINE 'SAME', XL1, TLX
      IF ( P_PRIQTY.1 \= P_N_PRIQTY.1 ) THEN DO
         TLX = 'PRIQTY CHANGED FROM' P_PRIQTY.1 'TO' P_N_PRIQTY.1
         CALL WRITE_MSG TLX, XL1
         END                        /* END IF PRIQTY \= N_PRIQTY     */
      /* IF PRIQTY LESS THAN SECQTY, WRITE A NOTE                    */
      IF ( P_N_PRIQTY.1 <  P_N_SECQTY.1 ) THEN DO
         TLX = 'PRIQTY LESS THAN SECQTY.  VERIFY'
         CALL WRITE_NOTE TLX, XL1
         END                        /* END IF PRIQTY \= SECQTY       */
      END                           /* END IF PRIQTY \= P_N_PRIQTY   */
   END                              /* END IF PRIQTYPOS > Ø          */

/*-------------------------------------------------------------------*/
/*  PARAGRAPH : UPDATE SECQTY VALUE IF NECESSARY.                    */
/*-------------------------------------------------------------------*/
IF ( SECQTYPOS > Ø ) THEN DO
   IF ( P_SECQTY.1 \= P_N_SECQTY.1 ) &,
      ( DDL_FUNCTION = 'CREATE' ) THEN DO
      'ISREDIT CURSOR = ' CL1 Ø
      'ISREDIT FIND SECQTY NEXT WORD'
      'ISREDIT (XL1,XC1) = CURSOR'
      'ISREDIT (LNX) = LINE' XL1
      PARSE VAR LNX TOKEN .
      'ISREDIT FIND "'SECQTY'" NEXT WORD'
      'ISREDIT FIND '''BLANK''' NEXT'
      'ISREDIT (XL2,XC2) = CURSOR'
      LNXTAIL = SUBSTR(LNX,XC2)
      IF ( WORDS(LNXTAIL) \= Ø ) THEN 'ISREDIT TSPLIT' XL2 XC2
      TLX = '     SECQTY ' P_N_SECQTY.1
      IF ( TOKEN \= 'SECQTY' ) THEN DO
         'ISREDIT TSPLIT' XL1 XC1-1
         CALL INSERT_LINE 'SAME', XL1+1, TLX
         END
      ELSE CALL INSERT_LINE 'SAME', XL1, TLX
      IF ( P_SECQTY.1 \= P_N_SECQTY.1 ) THEN DO
         TLX = 'SECQTY CHANGED FROM' P_SECQTY.1 'TO' P_N_SECQTY.1
         CALL WRITE_MSG TLX, XL1
```

```
         END                          /* END IF SECQTY \= P_N_SECQTY   */
      END                             /* END IF SECQTY \= P_N_SECQTY   */
   END                                /* END IF SECQTYPOS > Ø          */
RETURN               /* END OF SUB-ROUTINE UPDATE_INDEX_DDL.         */

/*********************************************************************/
/*  SUB-ROUTINE DDLERROR : ROUTINE TO REPORT ERRORS. IT ALSO FINDS   */
/*  WHERE TO INSERT MESSAGES IN THE FILE.                            */
/*********************************************************************/
/*  PARM STOPFLAG : INDICATE IF SR SHOULD TERMINATE EXECUTION OR NOT */
/*  PARM LINEPOS  : INDICATES WHERE TO START REPOSITIONING FOR MSG   */
/*  PARM KEYWORD1 : IS THE TOKEN TO SEARCH FOR FROM LINE LINEPOS     */
/*  PARM KEYWORD2 : IS THE 2ND TOKEN TO LOOK FOR AFTER LOOKING FOR   */
/*                  KEY1CNT TIMES FOR KEYWORD1 TOKEN.                */
/*  SEVERAL VARIABLES ARE EXPOSED TO THE PROCEDURE.                  */
/*********************************************************************/
DDLERROR: PROCEDURE EXPOSE SM LM CL1
ARG STOPFLAG, LINEPOS, KEYWORD1, KEYWORD2, KEY1CNT
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE DDLERROR BEGINS HERE.                                */
/*-------------------------------------------------------------------*/
SC_HEX_VAL = C2X(';')   /* SET LOCAL VAR TO AVOID NOT GETTING IT.    */
ZEDSMSG  = SM                                /* COPY SHORT MSG.   */
TEMPLMSG = LM                                /* COPY LONG MSG.    */
'ISREDIT CURSOR = ' LINEPOS Ø                /* REPOSITION CURSOR */
IF ( KEY1CNT = '' ) THEN KEY1CNT = 1         /* CHECK IF NULL VAL */
IF ( DATATYPE(KEY1CNT,'NUM') \= 1 ) THEN DO
   LM = 'ERROR ON DDLERROR ROUTINE. GOT A NON NUMERIC VALUE'
   SM = 'EDIT MACRO ERROR'
   'ISPEXEC SETMSG MSG(ISRZØØ1)'             /* SET ISPF MSG VARS */
   EXIT 8                                    /* ERROR IN THE PGM  */
   END                                       /* EXIT MACRO        */
DO I=1 TO KEY1CNT BY 1                       /* LOOP FIND N TIMES */
   IF ( KEYWORD1 = ';' ) THEN ,
        'ISREDIT FIND X"'SC_HEX_VAL'" NEXT'
   ELSE 'ISREDIT FIND "'KEYWORD1'" NEXT WORD'
   END                                       /* END OF LOOP       */
/* IF THE SECOND KEYWORD WAS SPECIFIED, PROCEED TO FIND IT.         */
IF ( KEYWORD2 \= '' ) THEN ,
   IF ( KEYWORD2 = ';' ) THEN ,
        'ISREDIT FIND X"'SC_HEX_VAL'" NEXT'
   ELSE 'ISREDIT FIND "'KEYWORD2'" NEXT WORD'
'ISREDIT (XLN1,XCC1) = CURSOR'               /* SAVE CURSOR POS   */
/* COMPOSE THE FINAL MESSAGE TO BE DISPLAYED BY PREFIXING W/A HEADER */
IF ( STOPFLAG = 'Y' ) THEN ,
   MSGTEXT = '**ERROR** ' || TEMPLMSG        /* BUILD THE MESSAGE */
IF ( STOPFLAG = 'N' ) THEN ,
   MSGTEXT = '*WARNING* ' || TEMPLMSG        /* BUILD THE NOTE    */
/* INSERT MSG NOTE BEFORE LINE WHERE ERROR WAS FOUND.               */
'ISREDIT LINE_BEFORE (XLN1) = MSGLINE (MSGTEXT)'
```

```
'ISREDIT CURSOR = ' XLN1 XCC1                    /* REPOSITION CURSOR */
IF ( STOPFLAG = 'Y' ) THEN DO
   ZEDLMSG = '   ' || TEMPLMSG                   /* SET 3 SPACES      */
   'ISPEXEC SETMSG MSG(ISRZØØ1)'                 /* SET ISPF MSG VARS */
   EXIT 8                                        /* EXIT THE EDIT MAC */
   END
RETURN                      /* END OF SUB-ROUTINE DDLERROR.          */

/*********************************************************************/
/*  SUB-ROUTINE WRITE_NOTE : SUB-ROUTINE TO WRITE AN ISPF NOTE       */
/*********************************************************************/
WRITE_NOTE: PROCEDURE
ARG NOTETEXT, LINEPOS
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE WRITE_NOTE BEGINS HERE.                              */
/*-------------------------------------------------------------------*/
NOTEOUT = '-->' NOTETEXT
'ISREDIT CURSOR = ' LINEPOS 1                    /* REPOSITION CURSOR */
'ISREDIT LINE_BEFORE (LINEPOS) = NOTELINE (NOTEOUT)'
RETURN                      /* END OF SUB-ROUTINE WRITE_NOTE.        */

/*********************************************************************/
/*  SUBROUTINE WRITE_MSG :  SUB-ROUTINE FOR WRITING AN INFOLINE      */
/*********************************************************************/
WRITE_MSG: PROCEDURE
ARG WARNTEXT, LINEPOS
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE WRITE_MSG BEGINS HERE.                               */
/*-------------------------------------------------------------------*/
WARNOUT = '--> INFO:' WARNTEXT
'ISREDIT CURSOR = ' LINEPOS 1                    /* REPOSITION CURSOR */
'ISREDIT LINE_BEFORE (LINEPOS) = MSGLINE (WARNOUT)'
RETURN                      /* END OF SUB-ROUTINE WRITE_MSG.         */

/*********************************************************************/
/*  SUBROUTINE REPOSITION_CURSOR:  NO PARAMETERS.                    */
/*********************************************************************/
REPOSITION_CURSOR: PROCEDURE EXPOSE CL1 CR1 SC_HEX_VAL
/*-------------------------------------------------------------------*/
/*  SUB-ROUTINE REPOSITION_CURSOR BEGINS HERE.                       */
/*-------------------------------------------------------------------*/
'ISREDIT CURSOR = ' CL1 Ø                       /* REPOSITION CURSOR  */
'ISREDIT FIND X"'SC_HEX_VAL'" NEXT'             /* FIND NEXT ; TOKEN  */
RETURN                      /* END OF SUB-ROUTINE REPOSITION CURSOR  */

/*********************************************************************/
/*  SUB-ROUTINE : COMPUTE_DASD_PARMS                                 */
/*********************************************************************/
/*   THIS SUBROUTINE WILL TAKE IN CONSIDERATION THE DEVICE TYPE, THE */
/*   DEVICE GEOMETRY AND THE USER SPECIFIED PERCENTAGE OF MAXIMUM    */
```

```
/*   AVAILABLE SPACE ON A VOLUME TO COMPUTE A MAXIMUM ALLOWABLE      */
/*   ALLOCATION VALUE.  THIS VALUE WILL FORCE PRIMARY AND SECONDARY  */
/*   ALLOCATIONS TO BE LESS OR EQUAL TO IT.                          */
/*********************************************************************/
COMPUTE_DASD_PARMS: PROCEDURE EXPOSE YVAL DFDASD DFPERC DEVTYPE,
      ZDATE TRKCAP CYLCAP TRKPCYL CYLPDEV DEVCAP USERCAP PARMOKAY,
      MIN_DASD_ALLOC
/*-------------------------------------------------------------------*/
/*  SUBROUTINE COMPUTE_DASD_PARMS BEGINS HERE.                       */
/*-------------------------------------------------------------------*/

/*-------------------------------------------------------------------*/
/*  PARAGRAPH : SET DEVICE TYPE TO BE USED FOR SPACE COMPUTATIONS.   */
/*-------------------------------------------------------------------*/
SELECT
   WHEN DFDASD =1 THEN DEVTYPE = '3380-J'     /* FOR 3380 MOD K - 1  */
   WHEN DFDASD =2 THEN DEVTYPE = '3380-KE'    /* FOR 3380 MOD K3- 2  */
   WHEN DFDASD =3 THEN DEVTYPE = '3380-K'     /* FOR 3380 MOD K - 3  */
   WHEN DFDASD =4 THEN DEVTYPE = '3390-1'     /* FOR 3390 MOD 1      */
   WHEN DFDASD =5 THEN DEVTYPE = '3390-2'     /* FOR 3390 MOD 2      */
   WHEN DFDASD =6 THEN DEVTYPE = '3390-3'     /* FOR 3390 MOD 3      */
   WHEN DFDASD =7 THEN DEVTYPE = '3390-9'     /* FOR 3390 MOD 9      */
   OTHERWISE           DEVTYPE = '3390-3'     /* DEFAULT DASD        */
   END                                        /* END OF SELECT STMT  */

/*-------------------------------------------------------------------*/
/*  SELECT DEVICE PARAMETERS DEPENDING ON DEVICE TYPE.               */
/*-------------------------------------------------------------------*/
IF ( DEVTYPE = '3390-9' ) THEN DO
   TRKCAP  = 48                               /* 12 4K BLOCKS P/TRCK */
   TRKPCYL = 15                               /* # OF TRACKS/CYL     */
   CYLPDEV = 10017                            /* # OF CYLS ON 3390-9 */
   CYLCAP = TRKCAP * TRKPCYL                  /* CYL CAPACITY IN KB  */
   MIN_DASD_ALLOC = 'PRIQTY 48 SECQTY 48'     /* SET TO 1 3390 TRACK */
   END
IF ( DEVTYPE = '3390-3' ) THEN DO
   TRKCAP  = 48                               /* 12 4K BLOCKS P/TRCK */
   TRKPCYL = 15                               /* # OF TRACKS/CYL     */
   CYLPDEV = 3339                             /* # OF CYLS ON 3390-3 */
   CYLCAP = TRKCAP * TRKPCYL                  /* CYL CAPACITY IN KB  */
   MIN_DASD_ALLOC = 'PRIQTY 48 SECQTY 48'     /* SET TO 1 3390 TRACK */
   END
IF ( DEVTYPE = '3390-2' ) THEN DO
   TRKCAP  = 48                               /* 12 4K BLOCKS P/TRCK */
   TRKPCYL = 15                               /* # OF TRACKS/CYL     */
   CYLPDEV = 2226                             /* # OF CYLS ON 3390-2 */
   CYLCAP = TRKCAP * TRKPCYL                  /* CYL CAPACITY IN KB  */
   MIN_DASD_ALLOC = 'PRIQTY 48 SECQTY 48'     /* SET TO 1 3390 TRACK */
   END
IF ( DEVTYPE = '3390-1' ) THEN DO
```

```
   TRKCAP  = 48                                /* 12 4K BLOCKS P/TRCK */
   TRKPCYL = 15                                /* # OF TRACKS/CYL     */
   CYLPDEV = 1113                              /* # OF CYLS ON 3390-1 */
   CYLCAP = TRKCAP * TRKPCYL                   /* CYL CAPACITY IN KB  */
   MIN_DASD_ALLOC = 'PRIQTY 48 SECQTY 48'      /* SET TO 1 3390 TRACK */
   END
IF ( DEVTYPE = '3380-K' ) THEN DO
   TRKCAP  = 40                                /* 10 4K BLOCKS P/TRCK */
   TRKPCYL = 15                                /* # OF TRACKS/CYL     */
   CYLPDEV = 2655                              /* # OF CYLS ON 3380-K */
   CYLCAP = TRKCAP * TRKPCYL                   /* CYL CAPACITY IN KB  */
   MIN_DASD_ALLOC = 'PRIQTY 40 SECQTY 40'      /* SET TO 1 3380 TRACK */
   END
IF ( DEVTYPE = '3380-KE' ) THEN DO
   TRKCAP  = 40                                /* 10 4K BLOCKS P/TRCK */
   TRKPCYL = 15                                /* # OF TRACKS/CYL     */
   CYLPDEV = 1770                              /* # OF CYLS ON 3380KE */
   CYLCAP = TRKCAP * TRKPCYL                   /* CYL CAPACITY IN KB  */
   MIN_DASD_ALLOC = 'PRIQTY 40 SECQTY 40'      /* SET TO 1 3380 TRACK */
   END
IF ( DEVTYPE = '3380-J' ) THEN DO
   TRKCAP  = 40                                /* 10 4K BLOCKS P/TRCK */
   TRKPCYL = 15                                /* # OF TRACKS/CYL     */
   CYLPDEV = 885                               /* # OF CYLS ON 3380-J */
   CYLCAP = TRKCAP * TRKPCYL                   /* CYL CAPACITY IN KB  */
   MIN_DASD_ALLOC = 'PRIQTY 40 SECQTY 40'      /* SET TO 1 3380 TRACK */
   END

/*-------------------------------------------------------------------*/
/* COMPUTE USER DEVICE CAPACITY TO BE USED TO LIMIT THE MAXIMUM SIZE */
/* FOR PRIMARY AND SECONDARY SPACE COMPUTATIONS.                     */
/*-------------------------------------------------------------------*/
DEVCAP = CYLCAP * CYLPDEV                 /* DEVICE   CAPACITY IN KB */
TEMPNUM = ( DFPERC / 100 ) * DEVCAP       /* TEMPORARY NUMERIC FIELD */
USERCAP = TRUNC(TEMPNUM)                  /* GET INTEGER PART OF NUM */
RETURN

/*********************************************************************/
/*  SUB-ROUTINE : COMPSIZE                                           */
/*********************************************************************/
/*   THIS SUBROUTINE COORDINATES THE COMPUTATIONS FOR :              */
/*   1)  PRIQTY, SECQTY BASED ON DEVICE GEOMETRY,                    */
/*   2)  SEGSIZE, BASED ON PRIQTY AND SECQTY VALUES,                 */
/*********************************************************************/
/*   THE PARTS VALUE WILL DETERMINE THE MAXIMUM SIZE FOR THE DB2     */
/*   OBJECT.  A VALUE OF ZERO INDICATES IS IS NON-PARTITIONED, IN    */
/*   WHICH CASE, ONLY THE FIRST POSITION OF THE ARRAY WILL BE USED   */
/*   TO STORE VALUES.                                                */
/*********************************************************************/
COMPSIZE: PROCEDURE EXPOSE PARM1  P_PRIQTY. P_SECQTY. ,
```

```
            P_N_PRIQTY. P_N_SECQTY. P_SEGSIZE P_N_SEGSIZE ,
            TRKCAP CYLCAP TRKPCYL CYLPDEV DEVCAP USERCAP
ARG BPNAME, PARTS, SEGSZ,  PARTNO
/*-------------------------------------------------------------------*/
/*  SUBROUTINE COMPSIZE BEGINS HERE.                                 */
/*-------------------------------------------------------------------*/

/*-------------------------------------------------------------------*/
/* SET THE DB2 LIMITS FOR PRIQTY BASED ON THE PARTS VALUE.           */
/*-------------------------------------------------------------------*/
SELECT
  WHEN ( PARTS >   Ø & PARTS <= 16 )  THEN MAXPRI = 4194Ø4
  WHEN ( PARTS >  16 & PARTS <= 32 )  THEN MAXPRI = 2Ø97152
  WHEN ( PARTS >  32 & PARTS <= 64 )  THEN MAXPRI = 1Ø48576
  OTHERWISE                            MAXPRI = 2Ø97152
  END                                              /* END SELECT */

/*-------------------------------------------------------------------*/
/* SET LOCAL VARIABLES TO BE USED BY PROCEDURE COMPUTATIONS.         */
/*-------------------------------------------------------------------*/
PRIQTY   = P_PRIQTY.PARTNO               /* SET WITH CURRENT VALUE  */
SECQTY   = P_SECQTY.PARTNO               /* SET WITH CURRENT VALUE  */
NPRIQTY  = P_PRIQTY.PARTNO               /* SET WITH CURRENT VALUE  */
NSECQTY  = P_SECQTY.PARTNO               /* SET WITH CURRENT VALUE  */

/*-------------------------------------------------------------------*/
/* SET PARAMETERS TO USE FOR OBJECTS USING 4K BUFFERPOOLS.           */
/*-------------------------------------------------------------------*/
IF ( SUBSTR(BPNAME,1,5) \= 'BP32K') THEN DO      /* IT IS NOT A 32K  */
   IF ( PRIQTY < 12 ) THEN PRIQTY = 12     /* MINIMUM FOR  4K OBJECT */
   IF ( SECQTY < 12 ) & ( SECQTY \= Ø ) THEN SECQTY = 12
   MAXSEC = 131Ø68                        /* DB2 4K MAX SECONDARY QTY */
   PAGESIZE = 4                           /* SET 4K PAGE SIZE VALUE   */
   END

/*-------------------------------------------------------------------*/
/* SET PARAMETERS TO USE FOR OBJECTS USING 32K BUFFERPOOLS.          */
/*-------------------------------------------------------------------*/
IF ( SUBSTR(BPNAME,1,5) = 'BP32K' ) THEN DO
   IF ( PRIQTY < 96 ) THEN PRIQTY = 96     /* MINIMUM FOR 32K OBJECT */
   IF ( SECQTY < 96 ) & ( SECQTY \= Ø ) THEN SECQTY = 96
   MAXSEC = 131Ø4Ø                       /* DB2 32K MAX SECONDARY QTY */
   PAGESIZE = 32                         /* SET 32K PAGE SIZE VALUE   */
   END

/*-------------------------------------------------------------------*/
/*  COMPUTE NPRIQTY TO MAXIMIZE TRACK AND CYLINDER CAPACITY AND      */
/*  ENSURE ALLOCATION IN TRACK OR CYLINDER UNITS.                    */
/*-------------------------------------------------------------------*/
IF ( PRIQTY <= TRKCAP ) THEN NPRIQTY = TRKCAP
ELSE DO
```

```
   /* COMPUTE THE NUMBER OF TRACKS NEEDED.                          */
   TRKS    = PRIQTY % TRKCAP
   TRKSREM = PRIQTY // TRKCAP
   IF ( TRKSREM \= Ø ) THEN TRKS = TRKS + 1
   /* COMPUTE THE NUMBER OF CYLINDERS NEEDED.                       */
   CYLS    = PRIQTY % CYLCAP
   CYLSREM = PRIQTY // CYLCAP
   IF ( CYLSREM \= Ø ) THEN CYLS = CYLS + 1
   /* COMPUTE THE NEW PRIMARY QUANTITY, NPRIQTY                     */
   /* IF THE NUMBER OF TRACKS IS LESS THAN 3 CYLS, ALLOCATE IN TRKS  */
   IF ( TRKS < ( TRKPCYL * 3 ) ) THEN NPRIQTY = TRKS * TRKCAP
   ELSE                               NPRIQTY = CYLS * CYLCAP
   END

/*-------------------------------------------------------------------*/
/*  VERIFY THAT PRIQTY IS LESS THAN THE MAXIMUM CAPACITY SPECIFIED   */
/*  BY THE USER ( DEVCAP * DFPERC / 1ØØ ) AND LESS THAN THE MAXIMUM  */
/*  SIZE ALLOWED BY DB2 AND DFDSP.  SET MAXSIZE TEMP VARIABLE TO BE  */
/*  THE SMALLEST VALUE BETWEEN MAXPRI & USERCAP.                     */
/*-------------------------------------------------------------------*/
IF ( MAXPRI > USERCAP )  THEN MAXSIZE = USERCAP
ELSE                          MAXSIZE = MAXPRI
IF ( NPRIQTY > MAXSIZE ) THEN NPRIQTY = MAXSIZE

/*-------------------------------------------------------------------*/
/*  COMPUTE NSECQTY TO MAXIMIZE TRACK AND CYLINDER CAPACITY.         */
/*  AND ENSURE ALLOCATION IN TRACK OR CYLINDER UNITS.                */
/*-------------------------------------------------------------------*/
IF ( SECQTY <= TRKCAP ) & ( SECQTY \= Ø ) THEN NSECQTY = TRKCAP
ELSE DO
   /* COMPUTE THE NUMBER OF TRACKS NEEDED.                          */
   TRKS = SECQTY % TRKCAP
   TRKSREM = SECQTY // TRKCAP
   IF ( TRKSREM \= Ø ) THEN TRKS = TRKS + 1
   /* COMPUTE THE NUMBER OF CYLINDERS NEEDED.                       */
   CYLS = SECQTY % CYLCAP
   CYLSREM = SECQTY // CYLCAP
   IF ( CYLSREM \= Ø ) THEN CYLS = CYLS + 1
   /* COMPUTE THE NEW SECONDARY QUANTITY, NSECQTY                   */
   /* IF THE NUMBER OF TRACKS IS LESS THAN 3 CYLS, ALLOCATE IN TRKS  */
   IF ( TRKS < ( TRKPCYL * 3 ) ) THEN NSECQTY = TRKS * TRKCAP
   ELSE                               NSECQTY = CYLS * CYLCAP
   END

/*-------------------------------------------------------------------*/
/*  VERIFY THAT SECQTY IS LESS THAN DB2 ALLOWED VALUE AND LESS THAN  */
/*  PRIQTY. IF NOT, SET TO HIGHEST OF THE TWO VALUES.                */
/*-------------------------------------------------------------------*/
IF ( MAXSEC > USERCAP ) THEN MAXSIZE = USERCAP
ELSE                         MAXSIZE = MAXSEC
IF ( NSECQTY > MAXSIZE ) THEN NSECQTY = MAXSIZE
```

```
/*-------------------------------------------------------------------*/
/*  FOLLOWING PARAGRAPH COMPUTES OPTIMUM SEGMENT SIZE FOR TABLESPACE */
/*  SEGMENT SIZE CAN GO FROM 4 TO 64 IN INCREMENTS OF 4.             */
/*-------------------------------------------------------------------*/
/*  SEGMENT SIZE IS THE NUMBER OF PAGES (OF X SIZE) THAT WILL BE     */
/*  USED TOGETHER BY DB2 TO STORE DATA FOR ONE OF POSSIBLE MANY      */
/*  TABLES INSIDE A SINGLE TABLESPACE.                               */
/*-------------------------------------------------------------------*/
/*  A 4K TABLESPACE WITH A SEGMENT SIZE OF 64K WILL HAVE A 256K      */
/*  CHUNK OF CONTIGUOUS DASD STORAGE TO BE USED BY A TABLE.          */
/*  A 32K TABLESPACE WITH A SEGMENT SIZE OF 64 WILL HAVE A 1M CHUNK  */
/*  OF CONTIGUOUS DASD STORAGE TO BE USED BY A TABLE.                */
/*-------------------------------------------------------------------*/

IF ( SEGSZ > Ø ) THEN DO                /* FOR SEGMENTED TABLESPACES */
   TSEGSZ = 64                          /* START AT  LARGEST SEGSIZE */
   /*----------------------------------------------------------------*/
   /* SET THE SEGSIZE VALUE DEPENDING ON DEVICE GEOMETRY AND NPRIQTY */
   /* IF NPRIQTY NOT A MULTIPLE OF SEGSIZE, TRY A NEW SEGSIZE VALUE  */
   /*----------------------------------------------------------------*/
   DO WHILE ( ( TSEGSZ > 4 ) &,
              ( ( NPRIQTY // ( TSEGSZ * PAGESIZE ) ) \= Ø ) )
      TSEGSZ = TSEGSZ - 4
      END                               /* END OF DO WHILE LOOP      */
   /* IF ( ( TSEGSZ*PAGESIZE ) > NPRIQTY ) THEN TSEGSZ = TSEGSZ - 4  */
   /*----------------------------------------------------------------*/
   /* SAVE NEW SEGSIZE VALUE FOR LATER DDL EDITING.                  */
   /*----------------------------------------------------------------*/
   P_N_SEGSIZE = TSEGSZ                     /* STORE NEW VALUE       */
   END                                      /* END IF SEGSZE > Ø     */

/*-------------------------------------------------------------------*/
/* STORE REVISED PRIMARY AND SECONDARY QUANTITIES IN ARRAYS.         */
/* STORE NEW PRIMARY AND SECONDARY QUANTITIES IN THE ARRAYS.         */
/*-------------------------------------------------------------------*/
P_PRIQTY.PARTNO = PRIQTY
P_SECQTY.PARTNO = SECQTY
P_N_PRIQTY.PARTNO = NPRIQTY
P_N_SECQTY.PARTNO = NSECQTY

/*-------------------------------------------------------------------*/
/* IF PARM1 IS SHOW THEN PROCEED TO PRINT THE ARRAY VALUES.          */
/*-------------------------------------------------------------------*/
IF (PARM1 = 'SHOW') THEN CALL PRNTPART PARTNO
RETURN

/*********************************************************************/
/* SUB-ROUTINE : PRNTPART.   PARM PASSED IS SIZE OF ARRAYS.          */
/* DEBUGGING SUB-ROUTINE TO DISPLAY INFORMATION ABOUT PARTS SIZES    */
/*********************************************************************/
PRNTPART:
```

```
ARG PRTCNT
/*--------------------------------------------------------------------*/
/*  SUB-ROUTINE PRNTPART BEGINS HERE.                                 */
/*--------------------------------------------------------------------*/
TCNT = 1
DO TCNT=1 TO PRTCNT BY 1
   SAY 'PART     ' TCNT
   SAY 'USE TYPE ' P_STOR_TYPE.TCNT
   SAY 'USING    ' P_STOR_NAME.TCNT
   SAY 'PRIQTY   ' P_PRIQTY.TCNT
   SAY 'SECQTY   ' P_SECQTY.TCNT
   SAY 'N_PRIQTY ' P_N_PRIQTY.TCNT
   SAY 'N_SECQTY ' P_N_SECQTY.TCNT
   SAY 'ERASE    ' P_ERASE.TCNT
   SAY 'FREEPAGE ' P_FREEPAGE.TCNT
   SAY 'PCTFREE  ' P_PCTFREE.TCNT
   SAY 'VALUES   ' P_VALUES.TCNT
   CALL PAUSEPRT
   END
RETURN                     /* END OF SUB-ROUTINE PRNTPART.          */

/*********************************************************************/
/*  SUB-ROUTINE : PAUSEPRT.  USED FOR DEBUGGING PURPOSES.            */
/*********************************************************************/
PAUSEPRT: PROCEDURE
/*--------------------------------------------------------------------*/
/*  SUB-ROUTINE PAUSEPRT BEGINS HERE.                                 */
/*--------------------------------------------------------------------*/
SAY 'PLEASE PRESS "ENTER" TO CONTINUE, OR "Q" TO EXIT MACRO'
PULL X
IF ( X = 'Q' | X = 'QUIT' ) THEN EXIT
RETURN                     /* END OF SUB-ROUTINE PAUSEPRT.          */

/*********************************************************************/
/*  SUB-ROUTINE : DISPLAY_DDLSTR. USED FOR DEBUGGING PURPOSES.       */
/*********************************************************************/
DISPLAY_DDLSTR: PROCEDURE EXPOSE DDLSTR
/*--------------------------------------------------------------------*/
/*  SUB-ROUTINE PAUSE PRINT BEGINS HERE.                              */
/*--------------------------------------------------------------------*/
TRACE OFF
DDLWORDS = WORDS(DDLSTR)
DO TCNT=1 TO DDLWORDS BY 1
   SAY 'WORD('||TCNT||') IS ' WORD(DDLSTR,TCNT)
   END
CALL PAUSEPRT
RETURN
```

DB2 thin workstations with Windows NT

Are you concerned about the footprint size of a DB2 Client or DB2 Connect Personal Edition (PE) on a workstation? Are you already putting your palm to your forehead at the thought of deploying a FixPack for either of these products across your enterprise? Trying to figure out a way to manage database connections for an entire sales force without big inconveniences? If any of these scenarios ring a bell,

you should consider the thin architecture offered for a DB2 Client or DB2 Connect Personal Edition running on a Windows 32-bit-based operating system.

You can install a DB2 Client or DB2 Connect PE on a workstation and have these workstations act as code servers to DB2 Thin-Client or DB2 Thin-Connect workstations in your enterprise. These thin workstations load the DB2 Client or DB2 Connect PE code across a LAN connection from these code servers. A thin workstation functions like any other DB2 Client or DB2 Connect PE workstation: this type of architecture is transparent to the user. The main difference is that the code is installed on a code server, and not individually on each workstation. Each thin workstation needs only a minimal amount of code and configuration to establish links to a code server. This is in contrast to a locally installed DB2 Client or DB2 Connect PE workstation, sometimes referred to as fat architecture, where all the code is stored and run locally. Don't confuse this configuration with a Citrix environment. In a Citrix environment, both the code and the processing is handled by the Citrix server, in a Thin environment no processing in done at the code server.

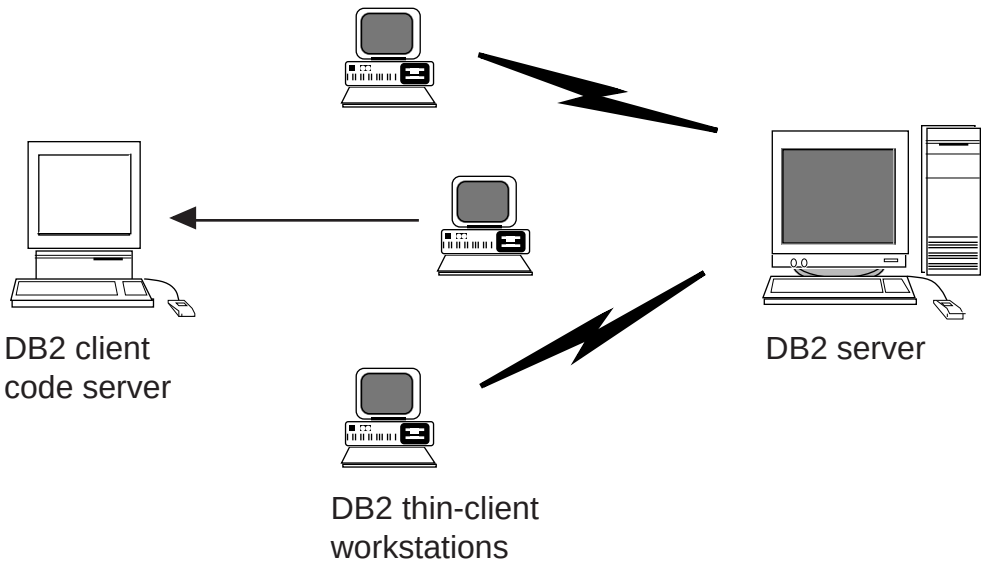

Figure 1: A typical DB2 thin-client set up

A typical DB2 Thin-Client environment is shown in Figure 1. A DB2 Administration Client is installed on a machine with the Thin Client Code Server component. After some configuration, this machine will be known as a DB2 Thin-Client Code Server. A DB2 Administration Client is the only type of client that can act as a code server for Thin-Client workstations. The DB2 Thin-Client workstations access the code server to dynamically load any code required. Once the code is loaded, all processing is done locally on the DB2 Thin-Client workstations. Using local database configuration information, a connection is made to a target DB2 server and the data is retrieved. It is important to understand that the DB2 code run on the Thin-Client workstations is loaded from the DB2 Thin-Client Code Server – there is no DB2 code installed on the Thin-Client workstations! This environment could also be extended to a DB2 server that is configured to access DB2 data residing on host or AS/400-based systems because it still uses the client-to-server data access model.

A typical DB2 Thin-Connect environment is shown in Figure 2. DB2 Connect PE is installed on a machine with the *Thin Connect Code Server* component. After some configuration, this machine will be

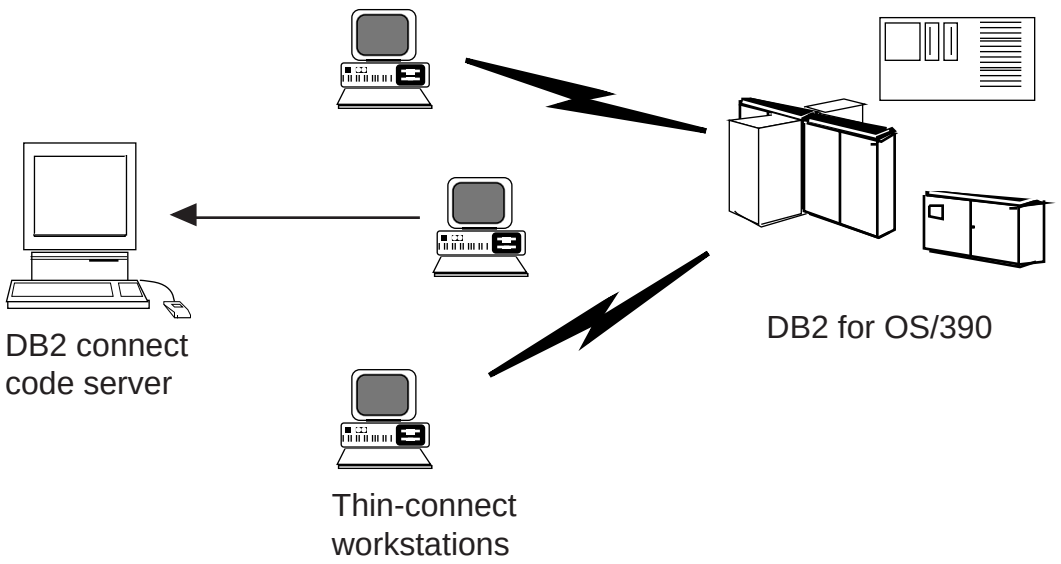

Figure 2: A typical DB2 thin-connect set up

known as a DB2 Thin-Connect Code Server. A DB2 Connect PE workstation is the only type of workstation that can act as a code server for DB2 Thin-Connect workstations. The DB2 Thin-Connect workstations function just like the DB2 Thin-Client workstations. They dynamically load any code required from the DB2 Thin-Connect Code Server. Once the code is loaded, all processing is done locally on the DB2 Thin-Connect workstations. Using local database configuration information, a connection is made to a target host or AS/400 DB2 server and the data is retrieved. In this example, the DB2 Connect code is run on the Thin-Connect workstations. This environment could also be extended to access a database on a DB2 server that is *not* located on a host or AS/400 system using the built-in client component with DB2 Connect PE.

This article will take you through the benefits and disadvantages of this type of architecture, the instructions necessary to set up this configuration, and some general tips on how to maintain this set-up. Throughout this article, I will assume that you are setting up a Windows NT (or Windows 2000) code server to service remote thin workstations that are on Windows NT (or Windows 2000) workstations. Any exceptions to this will be noted. Also, the term thin workstation will be used to represent either a DB2 Thin-Client or a DB2 Thin-Connect workstation, unless otherwise stated.

THE BENEFITS AND DISADVANTAGES OF A THIN ENVIRONMENT

As with most scenarios, there are advantages and disadvantages associated with a thin architecture in your environment. Ultimately, you have to go over these points and decide whether or not this type of configuration suits your business needs. For the most part, I think you will find this method of supporting a DB2 Client or DB2 Connect PE well suited for most IT business models.

Perhaps the greatest advantage of this type of set-up is that you install the code on only one machine. From a deployment and maintenance standpoint, this is a real treat. Imagine how easy it would be to roll out a new version of a DB2 Client or DB2 Connect PE – you would need to install or migrate the code on only one machine. This central-point-of-maintenance feature can also be applied to a rollout of a FixPack

as well; you would have to roll it out at only one location, yet all of your enterprise would have the updated code. This benefit also extends to that rare scenario where you have to rollback a FixPack as well. Once you have your code server installed, adding thin workstations is as simple as writing a script and firing it off in an e-mail. Your employees could receive the e-mail, and their workstations would become thin workstations. If you employed a system management tool in your environment, for example SMS, you could push these scripts out without any user interaction. The central point of maintenance and ease of creation helps to reduce the costs associated with the client-to-server business model since you do not need the IT staff to perform and support each and every installation.

Another benefit of this type of installation is the reduced footprint on each workstation. It is virtually none! The DB2 code for either DB2 product is installed only on the code server. There is no DB2 code that is installed on the thin workstation. This can be very beneficial for a sales force that uses a DB2 client or DB2 Connect PE edition on a laptop, where storage may be limited, to connect to DB2 data sources. (Keep in mind that the thin workstation must be connected to the network in order to access the code on the DB2 code server.)

Finally, the way you deploy this installation can follow a 'push' or 'pull' methodology. While you can write a script that your employees will 'pull', perhaps by receiving an e-mail, you can also integrate this type of deployment with a system management tool like SMS and 'push' the installation out to its target workstations.

As with anything great, there are always some disadvantages. You will be happy to know that there aren't many bad things associated with a thin workstation environment. You will experience a certain amount of performance degradation whenever a piece of code is loaded from the code server. Once this code has been loaded by the thin workstations, the fact that these are thin workstations becomes transparent to the user. Now, you should be aware that the thin workstations have to *dynamically* load libraries as needed, so if you start another DB2 application, you may experience a minor performance degradation period while this new application is being loaded. For example, let's suppose you have a Thin-Client. When you connect to a database, you will notice a short performance loss compared with a client where the

code is installed locally. After this code has loaded, your client will have the same performance as the locally installed client. If after you have connected to the database you decided to start the Client Configuration Assistant (CCA), you would then experience a performance loss as the code associated with this connectivity tool is loaded from the code server. In reality, you probably wouldn't be loading the CCA and other components very often, you would mostly be using the run-time environment. So it's not a very major drawback!

Another potential sticking point is the location of the catalog files. The catalog files contain all the information needed for a workstation to connect to a database. Currently, you have to maintain catalog files on each and every workstation, even if these are thin workstations – this is just like a regular installation. (There is one exception to this, if you are running IBM's Lightweight Directory Access Protocol (LDAP) which is supported in DB2 Version 6.) This may seem like a big problem, but I have a great solution for you, and things are going to get easier with the forthcoming Version 7 release of DB2. You can get around the pain of cataloguing databases on each and every thin workstation by using the profile export and import options provided by the CCA. A simple e-mail could easily be sent that would update each machine with the correct catalog information. Maintenance? No problem, just send another e-mail! With the added support for Microsoft's LDAP coming in DB2 Version 7, it's going to get even easier for you to deploy and maintain these thin workstations.

PERFORMING THE INSTALLATION OF A THIN-CONNECT CLIENT

Performing a Thin-installation is a straightforward process. To set up this type of environment, you need to perform the following steps:

- Install a DB2 Administration Client or DB2 Connect PE with the code server component.
- Set up cross-platform support on the code server (optional).
- Share the code server directory where the DB2 Client or DB2 Connect PE code is installed.
- Create a response file for the target thin workstation.
- Make the code server accessible to the target thin workstation.

- Create the target thin workstation.

Step 1 – install a DB2 Administration Client or DB2 Connect PE with the code server component

You need to install a DB2 Administration Client or DB2 Connect PE on the workstation that will act as the code server for the Thin-Client or Thin-Connect target workstations. A DB2 Thin-Client can load code only from a DB2 Thin-Client Code Server and a DB2 Thin-Connect workstation can load code only from a DB2 Thin-Connect Code Server.

To install either of these products, perform the following steps:

- Log-on to the system with a user account that belongs to the local Administrators group. For more information on the requirements to install these products, refer to the appropriate *Quick Start* manual.
- Shut down any programs that are running so that the set-up program can update files as required.
- Insert the appropriate CD-ROM into the drive. If you want to install a DB2 Thin-Client Code Server, make sure that you have inserted a CD-ROM that contains the DB2 Administration Client. A DB2 Administration Client is provided on any server CD-ROM or the DB2 Administration Client CD-ROM. If you want to install a DB2 Thin-Connect Code Server, you must insert the DB2 Connect Personal Edition CD-ROM.

  The auto-run feature automatically starts the set-up program. The set-up program will determine the system language, and launch the set-up program for that language. If you want to run the set-up program in a different language, or the set-up program fails to auto-start, invoke the set-up program manually.

  To manually invoke the set-up program, enter the ‘x:\setup /i language’ command, where ‘x:\’ represents your CD-ROM drive and ‘language’ represents the country code for your language (for example, EN for English).

  For a detailed list of all the available country codes, refer to the appropriate appendix in your *Quick Start* manual.

- The **Welcome** window opens. Click on the **Next** push button.
- The **Select Products** window opens. Select the product that you want to install (either a **DB2 Administration Client** or **DB2 Connect Personal Edition**).
- The **Select Installation Type** window opens. Click on the **Custom** graphic button.
- The **Select Components** window opens. Select the components that you want to install.

  If you want this workstation to act as a DB2 Thin-Client Code Server, you must select the **Thin Client Code Server** component. If you want this workstation to function as a DB2 Thin-Connect Code Server, you must select the **Thin Connect Code Server** component.
- Respond to the set-up program's remaining prompts. On-line help can walk you through the remaining steps of the installation process. You can invoke on-line help by clicking on the **Help** push button, or by pressing the F1 key, at any time.

  If you need more installation information, refer to the appropriate *Quick Beginning* manual.

Step 2 – set up cross-platform support on the code server (optional)

A code server can provide code only for thin workstations that belong to the same family of operating systems. This means that a Windows NT code server could not support a Windows 9*x*-based thin workstation and *vice versa*.

If you are not planning to support a mix of Windows NT and Windows 9*x* thin workstations in your environment, you can skip this step.

If you have deployed a heterogeneous configuration of Windows 32-bit operating systems in your enterprise, you can set up your code server to support both Windows NT-based and Windows 9*x*-based thin workstations by performing the following steps (note: the example that follows will assume that you are configuring a Windows NT-based code server to service Windows 9*x* thin workstations):

- Create a directory on the Windows NT code server that will be used to service Windows 9*x* thin workstations by entering the 'md d:\sqllib9x' command, where 'd:' represents a local hard drive.
- Copy the DB2 product directory on the code server (for example, c:\sqllib) into the directory that you just created by entering the following command:

  ```
  xcopy c:\sqllib\*.* d:\sqllib9x /s /e
  ```

  where 'c:' represents the drive on the code server where the DB2 product was installed, and 'd:' represents the drive on the code server where the sqllib9x directory was created in the previous step.
- Change focus to the directory that you created for the cross platform code server. For our example, you would enter the 'cd d:\sqllib9x' command. This directory would be used to serve thin workstations running on Windows 9*x*.
- To enable this machine to service a cross-platform thin workstation, enter the 'd:\sqllib9x\bin\db2thn9x.bat target_platform' command, where 'd:' is the local drive that you created to act as a code server for cross-platform thin workstations, and 'target_platform' is the platform that this directory will support. This value can take one of two settings – nt or 9*x*.

  If the code server was running Windows NT and you wanted it to service thin workstations running Windows 9*x*, you would use the 9*x* parameter. If the code server was running Windows 9*x* and you wanted it to service thin workstations running Windows NT, you would use the nt parameter.

  For our example, enter this command as follows:

  ```
  d:\sqllib9x\bin\db2thn9x.bat 9x
  ```

  Depending on the product you have installed, you may receive some 'File not Found' errors. You can ignore these. For some reason this script tries to copy files that may not be found on the DB2 product that you have installed.

You now have two code bases on your Windows NT code server.

If you are going to create a thin workstation on a Windows NT

machine, use the Windows NT code base (for example, c:\sqllib) in the following steps. If you are going to create a thin workstation on a Windows 9*x* machine and have the code served by a Windows NT code server, use the Windows 9*x* code base (for example, d:\sqllib9x) in the following steps.

There is one last thing to note if you are logged onto a Windows 9*x* Thin-Client workstation that is running code from a Windows NT code server: you must ensure that the user account that you are logged onto on the Windows 9*x* workstation is locally defined on the Windows NT code server.

Step 3 – share the code server directory where the DB2 Client or DB2 Connect PE code is installed

In order for the thin workstations to load the required code from the code server, each of the target thin workstations must be able to read the directory where the DB2 Client or DB2 Connect PE code is installed. To make the code directory available to all thin workstations in READ mode, perform the following steps:

- Click on **Start** and select **Programs/Windows NT Explorer**.
- Select the directory where you installed the DB2 product. For our example, use the c:\sqllib directory for thin workstations running on Windows NT. If you were going to set up thin workstations that were running on Windows 9*x*, you would also have to share the d:\sqllib9x directory.
- Select **File/Properties** from the menu bar.
- Select the **Sharing** tab.
- Select the **Shared As** radio button.
- In the **Share Name** field, enter a share name – for our example, NTCODESERVER.
- All target thin workstations need to have READ access to this directory for all users.

  If you are setting up a Windows NT-based code server, specify READ access for everyone as follows:

– Click the **Permissions** button. The **Access Through Share Permissions** window opens.

– In the **Name** box, select **Everyone**.

– Click on the **Type of Access** drop-down box and select **Read**.

– Click on **OK** until all windows are closed.

If you are setting up a Windows 9*x*-based code server, you do not need to specify this type of access when you set up a share. By default, everyone is granted read access.

Step 4 – create a response file for the target thin workstation

When you installed the code server, you performed an interactive installation. In this installation, you have to manually respond to prompts from the set-up program in order to install your product. Your responses provided the information needed to install the DB2 product and configure its environment. During a distributed installation, this information is provided in the form of keywords and values in a response file. For this reason, a distributed installation is often referred to as a response file installation or even a silent installation. For a detailed description of a distributed installation, refer to the *DB2 and DB2 Connect Installation and Configuration Supplement*. This on-line book will show you how to perform this type of installation for any DB2 product and also how to integrate this type of installation with a system management tool like SMS. For our example, I will give you the basics on how to perform this installation for a target thin workstation.

For a DB2 Thin-Client or DB2 Thin-Connect installation, there is a response file called db2thin.rsp, which you can use to install either thin workstation. The default settings for the most common installation type are provided in this file. You can find this response file in the c:\sqllib\thnsetup directory, where 'c:' represents the drive where you installed your DB2 product.

A response file contains:

- Keywords unique to installation.
- Registry value/environment variable settings.

- Database manager configuration parameter settings.

In a response file, the asterisk (*) acts like a comment. Any line that is prefixed with an asterisk will be ignored during the installation. If you do not specify a keyword, or it is commented out, a default value will be used. To enable a parameter, remove the asterisk. For example, let's assume that you want to install support for ODBC. The default entry for this keyword in the response file is:

```
*COMP          =ODBC_SUPPORT
```

To install the ODBC, you would remove the asterisk from this line so it would look like:

```
COMP           =ODBC_SUPPORT
```

For some keywords, you have to set values. To enable these keywords, remove the asterisk as usual, but make sure that you also replace the contents to the right of the equals sign with the value you want for that parameter. For example:

```
*DB2.DIAGLEVEL= Ø - 4
```

would be:

```
DB2.DIAGLEVEL = 4
```

if you wanted to set this parameter to 4.

The following is a section of the db2thin.rsp sample response file:

```
* Required Global DB2 Registry Variable
* -------------------------------------
DB2INSTPROF                    = C:\CFG


* General Options
* ---------------
*TYPE                    = Ø,1,2  (Ø=compact, 1=typical, 2=custom)
*COMP                    = ODBC_SUPPORT
*COMP                    = CONTROL_CENTER
*COMP                    = EVENT_ANALYZER
*COMP                    = WEB_ADMINISTRATION
*COMP                    = QUERYMONITOR
*COMP                    = TRACKER
*COMP                    = QUERYADMIN
*COMP                    = CLIENT_CONFIGURATION_ASSISTANT
*COMP                    = COMMAND_CENTER
*COMP                    = DOCUMENTATION
```

```
*CREATE_ICONS            = YES or NO (default=YES)
*REBOOT                  = YES or NO
```

Refer to the on-line supplement for detailed descriptions of these components and configuration parameters. For our example, set the following:

```
DB2INSTPROF        = C:\CFG
TYPE               = 1
REBOOT             = YES
```

Once you have finished editing this file, save it with a different name so that you can maintain the integrity of the sample. Call this file test.rsp and save it in the same directory that you shared in the previous step (c:\sqllib for our example).

Step 5 – make the code server accessible to the target thin workstation

The code server must be accessible before you can set up the target workstation to act as a thin workstation. From the target thin workstation, enter the net use command to attach the shared directory that you created on the code server as follows:

```
net use x: \\computer_name\directory_sharename /USER:domain\username *
```

where 'x:' represents the local drive used to connect to the remote shared directory; 'computer_name' represents the computer name of the code server; 'directory_sharename' represents the share name of the shared directory on the code server (in our example, this was NTCODESERVER); 'domain' represents the domain where the user account is defined (this is only required if the account is a domain account and you are not logged on to the system with a user account that has READ access on the remote code server); and 'username' represents a user that has access to this machine (this is only required if you are not logged on to the system with a user account that has READ access on the remote code server or you specified the domain parameter).

Note: if you are entering this command on a Windows 9*x* workstation, you must be logged on to the workstation as a valid user. The /USER parameter is not a valid option with this command on Windows 9*x*.

For example, to assign a shared directory called NTCODESERVER

on a machine called MYSERVER, to the x: drive, enter the following command:

```
net use x: \\MYSERVER\NTCODESERVER *
```

You may want to use the */P* option of the net use command to make this share permanent on your machine. For more information, refer to your Windows NT documentation for this command.

If you are planning on having a DB2 for Windows NT code server service a Windows 9*x* thin workstation (or *vice versa*), make sure that you enter the share name for the directory that you created for this purpose.

Step 6 – create the target thin workstation

To finish this set-up, you need to run the thnsetup command. This command will set up the DB2 Thin-Client or DB2 Thin-Connect workstation and the required links to the code server.

The thnsetup command can be entered with the following parameters:

```
drive:\path thnsetup /P drive:path\ /U drive: path\responsefile /L
drive:path\logfile /M machine /S sharename
```

where:

- /P specifies the path where the DB2 code is installed on the code server. Specify this parameter using the drive that you attached to in the previous step. Remember, if this is a Windows NT code server and your target workstations are running Windows 9*x*, you have to specify the directory you set up for Windows 9*x*. This parameter is required.
- /U specifies the fully-qualified response file name. This parameter is required. For our example, use the response file that you saved as test.rsp on the remote code server.
- /L specifies the fully-qualified log file name, where set-up information and any errors occurring during set-up are logged. If you do not specify the log file's name, the default db2.log file name is used. This file will be created in a directory called db2log, on the drive where your operating system is installed. This

parameter is optional. For our example, do not specify this parameter.

- /M specifies the computer name of the code server. This parameter is required.
- /S specifies the share name of the code server drive and directory where the DB2 product is installed. If you specify this parameter, you will not have to perform a net use command to the code server each time you reboot your machine. You do not need this command if you used the /P option when you performed the net use command in the previous step.

  If this share requires that a user account be entered, specify the share name as 'sharename,userid,password' (you may have to specify a domain name if you are a domain user). This parameter is optional. For our example, do not specify this parameter.

For example, to create a thin workstation where the shared NTCODESERVER directory on a code server called myserver is mapped locally to the 'x:\' drive, and the response file is called test.rsp and is located in the same directory as the code server, enter the following command:

```
x:\thnsetup\thnsetup /P x: /U x:\thnsetup\test.rsp /M MYSERVER
```

When the thnsetup command completes, check the messages in the log file. Since we set the machine to reboot once the set-up of the thin workstation completes, you will know you have encountered an error if your machine does not reboot.

Note: I have noticed that even without specifying the /P option with the net use command or the /S option with the thnsetup command, the workstation reboots with this share. If this works for you, that's great. This is not an officially supported feature of this command.

WHERE DO YOU GO FROM HERE?

Now that you have set up and configured your thin workstations, you need to set up access to the databases that you want your users to access on each workstation. I suggest you use the Client Configuration

Assistant (CCA) to set up access to these databases on the code server. This is a standard DB2 task, so I will refer you to your *Quick Start* manual for information on how to use this tool and its functions.

After you have catalogued all the databases on the code server that you want your thin workstations to access, you need to update all the local catalog directories on each thin workstation. Before you go out and catalog these database connections, allow me to leave you with some hints and tips to make this task easier to accomplish and maintain in the future.

I suggest you use the profile option of the CCA. The CCA allows you to export and import Client Profiles that contain database connection information, as well as configuration settings. On the code server, export a Client Profile. This profile will contain all the information needed to set up exactly the same database connections and configurations on each of the thin workstations that you just installed. Your end users can either use the CCA import option if you are comfortable with that, or if you are trying to hide DB2 from your users, you can use the db2cfimp command. This command could be pulled down via an e-mail or pushed out with SMS. Using this command allows you to hide any DB2 tasks from your end users. For more information on the db2cfimp command, refer to the *Command Reference*.

Finally, you should be aware that DB2 Universal Database Version 7 will support the Lightweight Directory Access Protocol (LDAP) solution from Microsoft that will be the standard method for maintaining this type of information on Windows systems. Currently, in Version 6, DB2 supports IBM's LDAP solution. So if you are using IBM's LDAP solution, you can take advantage of this central set-up and maintenance feature for your catalog directories right away.

I hope this article has introduced or clarified an exciting technology that you can use with DB2. If you follow the instructions and advice in this article, you should look forward to hassle-free DB2 Client and DB2 Connect PE deployments and updates!

Paul Zikopoulos
Software Engineer
IBM Canada (Canada)

Allen Systems Group has unveiled Version 5.2.2 of its ASG-IMPACT service management solution enabling broad customer support in distributed IT environments. It integrates service desk, problem, change, asset, and service level agreement management.

Automation capabilities accommodate existing network managers, job schedulers, performance monitors, frameworks, and auto-discovery tools.

The latest release supports the DB2 Universal Database Version 6.1 DB2 logon option plus MERANT Net Express Version 3.0.

For sites that run both ASG-IMPACT MVS and ASG-IMPACT NT, it's now possible to port MVS code directly to the ASG-IMPACT NT administrator's workstation.

For further information contact:
Allen Systems Group, 750 11th Street South, Naples, FL 34102, USA.
Tel: (941) 435 2200.
URL: http://www.asg.com.

* * *

IBM has announced the DB2 OLAP Server for OS/390, extending the DB2 OLAP Server OLAP engine to run on the System/390.

It can use either DB2 for OS/390 software to maintain its relational data, allowing the use of SQL programs and other relational tools to access and manage the data, or the Essbase multidimensional store can be used when higher performance is required.

DB2 Windows clients are connected to the OS/390 Server via TCP/IP.

It runs on Unix System Services of OS/390 and has the same functions as the workstation product. The components and add-on tools available with the workstation product, DB2 OLAP Server Version 1.1, can be used with the OS/390 version. Both are based on Hyperion Essbase 5.0.2.

For further information contact your local IBM representative.
http://www.software.ibm.com/data/db2.

* * *

IBM, i2, and Ariba have announced an alliance to deliver an end-to-end system for B2B e-commerce and collaboration.

In addition, both Ariba and i2 will extend their software to work with IBM's, including WebSphere, WebSphere Commerce Suite, DB2, and MQSeries, and will optimize the software for IBM's servers.

For further information contact your local IBM representative.
http://www.software.ibm.com/data/db2.

* * *