108 DB2

October 2001

update

DB2 Update

Published by

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

North American office

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

Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs £255.00 in the UK; $380.00 in the USA and Canada; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1997 issue, are available separately to subscribers for £22.50 ($33.75) each including postage.

DB2 Update on-line

Code from *DB2 Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/db2update; you will need to supply a word from the printed issue.

Editor

Trevor Eddolls

Disclaimer

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

Contributions

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

Printed in England.

Resolving RI constraints when moving data between mid-range machines

A common DBA task is to copy data from a database on one operating system (say Unix) to a database on a different one (say NT). You cannot restore an image copy across operating systems. The only option open to you is to use the DB2LOOK and DB2MOVE commands. Briefly, DB2LOOK gives you the DDL of the database/ tablespaces/tables etc, and DB2MOVE unloads the tables in IXF format. You then move all the files generated by the two commands to the receiving operating system, and run the reverse jobs to DB2LOOK and DB2MOVE. But what happens if you have referential integrity (RI) between the original tables? You have to load the parent tables before the child tables, otherwise you will get RI errors! There are products on the market that will do this for you, but here is a simple piece of SQL and some REXX to take the pain out of deciding in which order to load the tables.

The following was tested on Windows NT4 (SP5) and DB2 UDB 7.2.

On the receiving database you have to run the output from the DB2LOOK command to create the objects (and the RI), and then load the data that was unloaded using the DB2MOVE command. To determine the order in which to load the tables:

1 Run the following SQL (SQL1):

```
select max(length(tabname)) from syscat.tables
```

The output from this query is used as one of the inputs (<*table-length-name*>) to SQL2 below.

2 Run the following SQL (SQL2):

```
select
substr(tabname,1,<table-name-length>) as "tabxxxname",
parents,children,selfrefs,const_checked
from syscat.tables
where
(parents > 0 or children > 0)
and tabschema = '<the-table-schema>'
order by 2,3;
```

where:

– <*table-length-name*> is the answer from SQL1.

– <*the-table-schema*> is your table schema.

This will produce something like:

```
tabxxxname PARENTS CHILDREN SELFREFS CONST_CHECKED
```

It is a quick way to see how many parents/children each table has, and it will give you a rough idea of how complicated a task the load process will be.

Save the output in a text file called *in01.txt*. Edit this file, so that it contains just the raw data and not the header data ie:

```
TABLE_01     0    1    0 YYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
TABLE_02     0    2    0 YYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
TABLE_03     0    2    0 YYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
TABLE_04     0    4    0 YYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
```

3 Run the following SQL (SQL3).

Use the following SQL to get the maximum column length of the columns needed to list out the RI constraint names.

```
select
max(length(a.constname)),
max(length(a.reftabschema)),
max(length(a.reftabname)),
max(length(a.refkeyname)),
max(length(a.tabschema)),
max(length(a.tabname)),
max(length(b.colname))
from syscat.references a, syscat.keycoluse b
where a.constname = b.constname
and a.tabschema = b.tabschema
and a.tabname = b.tabname;
```

You should get back something like:

```
1          2          3          4          5          6          7
---------.----------.----------.----------.----------.----------.----------
        18          8         15         18          8         18         15
```

Plug the above values into the SQL in item 4.

4 This is SQL4.

Run the query below to get a list of all parent/children table names.

```
select
substr(a.constname,1,<val1>) as "Constraint",
substr(a.reftabschema,1,<val2>) as "Parent__sh",
substr(a.reftabname,1,<val3>) as "Parent_nm",
substr(a.refkeyname,1,<val4>) as "Parent_col",
substr(a.tabschema,1,<val5>) as "Child_sh",
substr(a.tabname,1,<val6>) as "Child_nm",
substr(b.colname,1,<val7>) as "Child_col"
from syscat.references a, syscat.keycoluse b
where a.constname = b.constname
and a.tabschema = b.tabschema
and a.tabname = b.tabname
order by 3,6;

Constraint Parent__sh Parent_nm  Parent_col Child_sh  Child_nm
Child_col
```

Save the output in a text file called *in02.txt*. Edit this file, so that it contains just the raw data and not the header data, ie:

```
SQLXX DB2ADMIN   TABLEAA   SQLXX  DB2ADMIN   TABLEBB   COLAA
SQLXX DB2ADMIN   TABLECC   SQLXX  DB2ADMIN   TABLEDD   COLBB
SQLXX DB2ADMIN   TABLEEE   SQLXX  DB2ADMIN   TABLEFF   COLCC
SQLXX DB2ADMIN   TABLEGG   SQLXX  DB2ADMIN   TABLEHH   COLDD
```

The above is an example of how the output should look in the *in02.txt* file.

5 Now you come to the point where you have all the information you need to determine the correct load order so as not to get RI load errors. You can either use the tried and tested pen and paper method and draw out your table structures, or you can use the REXX below.

6 The REXX code.

The code assumes that the schema is the same for all tables.

There is a trace level parameter (itrace), which basically writes out the input/processing/output lines. For the first couple of runs, I would set itrace to 9 (as shown in the example), so that you can see what is happening.

```
/* To display the order in which tables have to be loaded to account for
```

```
RI. */

parse upper arg itrace born indfØ1 indfØ2

/* If you are redirecting the output to a file rather than to the
screen, put born = 1 */

/*********************************************************************/
/* indfØ1 - TABLEYY   Ø  1  Ø YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY      */
/* indfØ2 - SQLØ2 DB2ADMIN  TAB1 SQLØ1 DB2ADMIN   TAB2   COLA      */
/*                                                                   */
/*                                                                   */
/* itrace trace level:                                               */
/* -1 trace everything (ie trace all)                                */
/* >4 tells you what's happening in the section deciding             */
/* which table to process first.                                     */
/* >8 a high level of reporting.                                     */
/*                                                                   */
/*********************************************************************/
/* Assumptions made in the exec:                                     */
/* (a) That the schema is the same for all tables                    */
/* (ie the schema is NOT checked).                                   */
/*********************************************************************/
/* Read in both input files.                                         */
/* db.1.xx will contain the data from indfØ1.                        */
/* db.2.xx will contain the data from indfØ2.                        */
/*********************************************************************/

indf.1 = indfØ1
indf.2 = indfØ2
Do jk = 1 to 2
  rc=stream(indf.jk,'c','query exists')
  If rc='' then Do
    say 'File:' indf.jk 'was not found!'
       exit
  End /* If rc='' then Do */
  Else Do
    file=.stream~new(indf.jk)
    lines=file~makearray(line)
    xl.jk = Ø
    Do linn over lines
      xl.jk = xl.jk + 1
      xx = xl.jk
      db.jk.xx = linn
    End /* do */
  End /* If rc='' then Do */
End /* Do jk = 1 to 2 */

/* xl.1 is the total number of lines read in from file1. */
/* xl.2 is the total number of lines read in from file2. */
```

```
If(itrace = -1) then Do
  trace all
End /* If(itrace = -1) then Do */

/*********************************************************************/
/* File2 contains a list of the parents and children.                */
/*********************************************************************/

outcx. = Ø
outpx. = Ø

/* Read in file 2 */
Do jk = 1 to xl.2
  par2.jk = subword(db.2.jk,3,1)
  chd2.jk = subword(db.2.jk,6,1)
  If(itrace > 8) then Do
    say "parent is" par2.jk "child is" chd2.jk
  End /* If(itrace = 2) then Do */
End /* Do jk = 1 to tinØ2 */

xt = Ø /* number of tables processed from file1. */

lx = Ø
Do jk = 1 to xl.1
  tab1 = subword(db.1.jk,1,1)
  If(length(tab1) > lx) then Do
    lx = length(tab1)
  End /* If(length(tab1) > lx) then Do */
End /* Do jk = 1 to xl.1 */

If(born = 1) then Do
  Nop
End /* If(born = 1) then Do */
Else Do
  say "The maximum table name length is:" lx
  say "Do you want to change the maximum table name length? (y/N)"
  parse upper pull ans
  If(ans = "Y") then Do
    say "Enter the maximum table length name ===>"
    parse upper pull ans
    If(ans = " ") then ans = lx
    lx = ans
    say "Continuing with a maximum table name length of:" lx
  End /* If(ans = 'Y') then Do */
End /* If(born = 1) then Do */

/* Read in file1 */
Do jk = 1 to xl.1
  tab1 = subword(db.1.jk,1,1)
```

```
  par1 = subword(db.1.jk,2,1)
  chd1 = subword(db.1.jk,3,1)
  If(itrace > 8) then Do
    say "table is" tab1 "parent is" par1 "child is" chd1
  End /* If(itrace = 2) then Do */

  xt = xt + 1
  outtn.xt = tab1
  outpn.xt.Ø = ' '
  outcn.xt.Ø = ' '
  outpx.xt = Ø

  If(chd1 > Ø) then Do
  /* A child/children exist(s) for the table. */
    xx = Ø
    ic = 1
    Do jk2 = 1 to chd1
      iflag1 = Ø
      Do jk3 = ic to xl.2
        If(itrace > 8) then Do
          say tab1 par2.jk3
        End /* If(itrace = 2) then Do */
        If(tab1 = par2.jk3) then Do
          ic = jk3 + 1
          iflag1 = 1
          temp1 = outcn.xt.xx
          xx = xx + 1
          outcn.xt.xx = left(chd2.jk3,lx,' ')
          outcx.xt = xx
          If(itrace > 4) then Do
            say "outcn " xt xx "=" outcn.xt.xx
            say "Child " left(xx,3) left(chd2.jk3,lx) "found for table "
left(tab1,lx)
          End /* If(itrace > 4) then Do */
          leave jk3
        End /* If(tab1 = par2.jk3) then Do */
      End /* Do jk3 = 1 to tinØ2 */
      If(iflag1 = Ø) then Do
        say "No child found for parent" tab1
      End /* If(iflag1 = Ø) then Do */
    End /* Do jk2 = 1 to chd1 */
  End /* If(chd1 > Ø) then Do */

  If(par1 > Ø) then Do
  /* A parent(s) exist(s) for the table */
    xx = Ø
    ip = 1
    Do jk2 = 1 to par1
      iflag1 = Ø
      Do jk3 = ip to xl.2
```

```
        If(tab1 = chd2.jk3) then Do
          ip = jk3 + 1
          iflag1 = 1
          xx = xx + 1
          outpn.xt.xx = left(par2.jk3,lx,' ')
          outpx.xt = xx
          If(itrace > 4) then Do
            say "Parent" left(xx,3) left(par2.jk3,lx) "found for table "
left(tab1,lx)
          End /* If(itrace > 4) then Do */
          leave jk3
        End /* If(tab1 = chd2.jk3) then Do */
      End /* Do jk3 = 1 to tinØ2 */
      If(iflag1 = Ø) then Do
        say "No parent found for child" tab1
      End /* If(iflag1 = Ø) then Do */
    End /* Do jk2 = 1 to par1 */
  End /* If(par1 > Ø) then Do */
End /* Do jk = 1 to tinØ1 */

/*********************************************************************/
/* write out the full results. */
/*********************************************************************/
/* Set maxnp - the maximum number of parents for any table. */
maxnp = Ø
Do jk = 1 to xt
  If(outpx.jk > maxnp) then Do
    maxnp = outpx.jk
  End /* If(outpx.jk > maxnp) then Do */
End /* Do jk = 1 to xt */

/*********************************************************************/
/* Work out the header line. */
/*********************************************************************/
noep = (lx +1)  * maxnp
noep = noep + 1
noec = 132 - noep - lx -2
wp = copies("=",noep)
wn = (noep%2) - 4
wp = overlay(' Parents ',wp,wn)

wc = copies("=",noec)
wn = (noec%2) - 6
wc = overlay(' Children ',wc,wn)

til.1 = wp || "$"  || center("table",lx) || "$" || wc
say til.1

/*********************************************************************/
/* Build the result set. */
```

```
/*********************************************************************/
Do jk = 1 to xt
  xxc = ' '
  Do jk2 = 1 to outcx.jk
    xxc = xxc  || left(outcn.jk.jk2,lx,' ') || ' '
  End /* Do jk2 = 1 to outcx.jk */
  xxp = ' '
  Do jk2 = 1 to outpx.jk
    xxp = xxp  || left(outpn.jk.jk2,lx) || ' '
  End /* Do jk2 = 1 to outpx.jk */
  xx = maxnp - outpx.jk
  outb = copies(' ',lx)
  xxb = ''
  Do jk2 = 1 to xx
    xxb = xxb || outb || ' '
  End /* Do jk2 = 1 to xx */
  xxp = xxb || xxp
  xxl = xxp || "$" || left(outtn.jk,lx,' ') || "$" || xxc
  say xxl
End /* Do jk = 1 to xt */

/*********************************************************************/
/* We now want to list out the tables in loading order.*/
/*********************************************************************/

/* The rules are: */
/* If the table has no parent, then load the table. */
/* If the table is a child, then does it have any other parents? */
/* If no then restore child */
/* If yes, then have the other parents been restored? */
/* If yes then load the child */
/* If no then cycle */

say copies('*',132)
say "The number of tables to process is:" xt
say copies('*',132)

/* Set up porn - Table (P)rocessed or (N)ot yet. */
Do jk = 1 to xt
  porn.jk = 'N'
End /* Do jk = 1 to xt */

/* What we basically do is trundle around all the tables, until they
have all been processed. */
/* This may not be the most efficient way of doing it, but it works. */

linx = Ø
iflag4 = 1
jk = Ø
Do until iflag4 = Ø
```

```
  inp = 0
  ipp = 0
  Do jk3 = 1 to xt
    If(porn.jk3 = 'N') then Do
      inp = inp + 1
    End /* If(porn.jk3 = 'N') then Do */
    Else Do
      ipp = ipp + 1
    End /* If(porn.jk3 = 'N') then Do */
  End /* Do jk3 = 1 to xt */
  If(itrace > 4) then Do
    say "Have processed this many tables:" ipp
  End /* If(itrace > 4) then Do */

  If(ipp = xt) then Do
    iflag4 = 0
  End /* If(iflag5 = 1) then Do */
  Else Do
    iflag4 = 1
  End /* If(iflag5 = 1) then Do */

  jk = jk + 1
  If(jk > xt) then Do
    jk = 1
  End /* If(jk > xt) then Do */

  If(porn.jk = 'N') then Do
    If(itrace > 4) then Do
      say "Am starting processing for table:" outtn.jk "with this many
parents:" outpx.jk
    End /* If(itrace > 4) then Do */
    If(outpx.jk = 0) then Do
      linx = linx + 1
      linp.linx = outtn.jk
      porn.jk = 'P'
      If(itrace > 4) then Do
        say "Table processed:" outtn.jk
      End /* If(itrace > 4) then Do */
    End /* If(outpx.jk = 0) then Do */

    If(outpx.jk > 0) then Do
      iflag2 = 0
      Do jk2 = 1 to outpx.jk
        Do jk3 = 1 to xt
          If(outtn.jk3 = outpn.jk.jk2) then Do
            If(porn.jk3 = 'P') then Do
              iflag2 = iflag2 + 1
            End /* If(porn.jk = 'P') then Do */
            Else Do
              If(itrace > 4) then Do
```

```
                say "Table" outtn.jk3 "not yet processed for" outtn.jk
              End /* If(itrace > 4) then Do */
            End /* If(porn.jk = 'P') then Do */
          End /* If(outtn.jk3 = outpn.jk.jk3) then Do */
        End /* Do jk3 = 1 to xt */

        If(outpn.jk.jk2 = outtn.jk) then Do
          If(itrace > 4) then Do
            say "Self referencing table found:" outtn.jk
          End /* If(itrace > 4) then Do */
          iflag2 = iflag2 + 1
        End /* If(outpn.jk.jk2 = outtn.jk) then Do */

      End /* Do jk2 = 1 to outpx.jk */

      If(iflag2 = outpx.jk) then Do
        /* All parents already loaded - therefore load the child. */
        linx = linx + 1
        linp.linx = outtn.jk
        porn.jk = 'P'
        If(itrace > 4) then Do
          say "Table processed:" outtn.jk
        End /* If(itrace > 4) then Do */
      End /* If(iflag2 = outpx.jk) then Do */
    End /* If(outx.jk > 0) then Do */
  End /* If(porn.jk = 'N') then Do */
End /* Do until iflag4 = 0 */

say "The tables should be loaded in the order shown below.  First is
top:"
Do jk = 1 to linx
  say linp.jk
End /* Do jk = 1 to linx */

Exit
```

7 I ran the above using ObjectREXX for Windows as follows:

```
>rexx ri01.txt 9 0 in01.txt in02.txt
```

The code was written to output to screens 132 characters wide, and you should see the following:

```
The maximum table name length is: 10
Do you want to change the maximum table name length? (y/N)

================= Parents =====
==============$  table   $============================== Children
========================
*****************************************************************
```

```
The number of tables to process is: 23
**********************************************************************
```

The tables should be loaded in the order shown below. First is top:

```
TABLEAA
TABLEBB
TABLECC
```

The above output will show the parent (if any) and child relationship for each table. It will also tell you an order in which to load the tables (ie parents first). It might not be the most efficient way to load the tables, but it will load them without any RI errors.

9 This completes the process. You now have the order in which the tables need to be loaded.

10 You then have to edit the *db2move.lst* file to reflect the correct table load order before processing it.

I have run this for about 30 tables, with a maximum of six parents and children.

C Leonard
Freelance Consultant (UK)

DB2 SYSCOPY management

INTRODUCTION

This article is an implementation of the *ImageCopy generator procedure* (August 2001, Issue 106) and *Recover tablespace* (September 2001, Issue 107) articles, already published in *DB2 Update*. This function, *DB2 SYSCOPY management*, allows users to carry out the display of the information for the recovery of tablespaces and run the MODIFY utility. MODIFY should be run regularly to clear outdated information from SYSIBM.SYSCOPY and SYSIBM.

SYSLGRNX. These tables, particularly SYSIBM.SYSLGRNX, can become very large and take up considerable amounts of space. By deleting outdated information from these tables you can help improve performance for processes that access data from them.

PROCEDURE DESCRIPTION

The tool executes a query on the DB2 catalog and displays the recovery information available for a specific tablespace or a group of tablespaces. At this point, it's possible to return to the previous panel or generate the MODIFY job to delete all the recovery information from the DB2 catalog.

CHECKLIST FOR INSTALLATION

Follow these steps to install the components of the REXX procedure.

Use the preallocated USER.LIBRARY – the USER.LIBRARY allocated for the *ImageCopy generator procedure.*

- Copy all REXXs, Macros, Help panels, and Panels into the USER.LIBRARY in the following members:
  - REXXs – $DB2TL0B, $DB2FOR0.
  - Panels – $DB2P000, $DB2P010, $DB2P011.
  - Help panels – $DB2H017, $DB2H018.
  - Source enhancement – IMPL00.
  - Macros – UPDATE00, $MDB2031.
- Run macro UPDATE00 on the original REXX source code $DB2TL00.
- Replace panel $DB2P000 with the new version.
- After installation, delete members IMPL00 and UPDATE00.

The test environment is DB2 Version 5 in OS/390 Version 8.

IMPL00 SOURCE CODE

```
when @db2opt = B then do      /*- DB2 SYSCOPY Managment -*/
             call #DB2TLØB parmpass
             @db2msg = result
             @db2opt =
             end
```

UPDATE00 MACRO UPDATE

```
/* REXX */
trace ?o
/*-----   Update Macro Enancement   -----*/
say ''
say '           #DB2TLØØ REXX update is running.......                '
say ''
isredit macro
isredit find ''IMPLØØ''
call VerRc     IMPLØØ
isredit copy   IMPLØØ after .zcsr
call VerRc     IMPLØØ
call Exlimpl   IMPLØØ
say ''
say '    +-------------------------------------------------------------+'
say '    |                                                             |'
say '    |      #DB2TLØØ REXX update successfully executed             |'
say '    |                                                             |'
say '    +-------------------------------------------------------------+'
say ''
isredit end
exit
VerRc :
   arg parmRc
   if rc > Ø then do
    say ' +-------------------------------------------------------------+ '
    say ' |  !!!!!!!!!!!!!   A T T E N T I O N     !!!!!!!!!!!!!   | '
    say ' |          Error during Code Update Function                  | '
    say ' |            Return Code 'rc' into Step 'parmRc'              | '
    say ' |   Activate trace ?r function and redo the UPDATEØØ Macro | '
    say ' |    on the original Source Code downloaded from Web Site  | '
    say ' +-------------------------------------------------------------+ '
      isredit find parmRc
      exit
      end
   return
Exlimpl :
   arg parmRc
   if rc = Ø then do
```

```
      isredit exclude parmRc all
      isredit delete x all
      isredit save
      end
   return
```

$DB2TL0B REXX EXEC

```
/* REXX */
trace ?o
   /*---       DB2  SYSCOPY  maintenance       ---*/
arg parmin
parm     = translate(parmin,' ',',')
nparm    = words(parm)
@db2subs = word(parm,1)
@db2ver  = word(parm,2)
   /*---  Parameters assignment                ---*/
 call @db2par0 @db2subs
 if word(result,1) = 99 then exit
 $lpar    =word(result,1) ;$accn   =word(result,2) ;$class
=word(result,3)
 $msgcla  =word(result,4) ;$region =word(result,5) ;$msglvl
=word(result,6)
 $notif   =word(result,7) ;$user   =word(result,8) ;$unitda
=word(result,9)
 $unitta  =word(result,10);$esunit =word(result,11);$prt
=word(result,12)
 $hiwork  =word(result,13);$db2ver =word(result,14);$ctsubs
=word(result,15)
 $librexx =word(result,16);$parmlib=word(result,17);$proclib
=word(result,18)
 $jcllib  =word(result,19);$report =word(result,20);$libexec
=word(result,21)
 $isptenu =word(result,22);$isppenu=word(result,23);$ispmenu
=word(result,24)
 $ispslib =word(result,25);$plilink=word(result,26);$sibmlnk
=word(result,27)
 $sortlib =word(result,28);$runlib =word(result,29);$dsnload
=word(result,30)
 $tep2pgm =word(result,31);$tep2pln=word(result,32);$unlopgm
=word(result,33)
 $unlopln =word(result,34);$dunlopg=word(result,35);$dunlopl
=word(result,36)
 $dsnproc =word(result,37)
/*---  Work areas initialization            ---*/
 blk   =   ; work = 0 ; wrk = 0 ; delon = off ; #sx = 0 ; cur00 =
'@db2db'
 call free
```

```
 do forever
    @db2data = date(e)
    @db2ora  = time()
    wrexit = ok
    address ispexec "display panel(@db2pØ1Ø) cursor("curØØ")"
    @db2msg =
    if rc = 8 then do
       if delon = on then
          call FreeØ
       curØØ   = '@db2subs'
       @db2opt =
       @db2msg =
       return  @db2msg
       end
    if @db2db = 'DSNDBØ4'  | @db2db = 'DSNDBØ6'  | ,
       @db2db = 'DSNDBØ7'  | @db2db = 'DSQ1STBB' | ,
       @db2db = 'DSNRGFDB' | @db2db = 'DSNRLST'  | ,
       @db2db = 'DSQDBCTL' | @db2db = 'DSQDBDEF'  then do
       @db2msg = 'Non } possibile trattare SYSTEM DataBase |||| '
       curØØ = '@db2db'
       wrexit = ko
       end
    if wrexit = ok then call SelDB2
    end
 exit
   /*---  DB2 Routine                          ---*/
 SelDB2 :
   /*---  File bridge name                     ---*/
    delon     = on
    outdstsin = $hiwork'.'@db2subs'.'$user'.SYSTSIN'
    outdsprt  = $hiwork'.'@db2subs'.'$user'.SYSPRINT'
    outdsrec  = $hiwork'.'@db2subs'.#DB2TLØB.SYSRECØØ'
   /*---  SYSIN file allocation                ---*/
    outdsin= $hiwork'.'@db2subs'.'$user'.SYSIN'
    dsnchk = sysdsn(''''outdsin'''')
    if dsnchk ¬= OK then do
       prmalloc = @db2subs' 'outdsin' Ø 5,1 f,b 8Ø 2792Ø sysin no'
       call @db2allØ prmalloc
       if word(result,1) = 99 then
          exit
       end
    else
       "alloc da('"outdsin"') f("sysin") shr reuse"
sk.1=' SELECT  TSNAME,ICTYPE,"TIMESTAMP"                                '
sk.2=' FROM SYSIBM.SYSCOPY                                              '
sk.3='      WHERE DBNAME =    '''@db2db'''                              '
    ldb2tbsp = length(@db2tbsp)
    silike = substr(@db2tbsp,ldb2tbsp,1)
    if silike ='%' | silike = '*' then do
```

```
       @db2tbsp = translate(@db2tbsp,'%','*')
sk.4='        AND TSNAME LIKE '''@db2tbsp'''                          '
       end
    else
sk.4='        AND TSNAME =    '''@db2tbsp'''                          '
sk.5='      ORDER BY TSNAME DESC ;                                    '
    sk.Ø=5;
    jobw = sysin
    call WriteRec
    call @db2acc1 @db2subs',#DB2TLØB'
    RCdb2  = word(result,1) /* DB2 access RC                 */
    okunlo = word(result,2) /* Access OK or Cursor Position  */
    NRrec  = word(result,3) /* Nr. Record or Message         */
   /*---  DB2 access OK start process          ---*/
    if RCdb2 = ØØ then do
       if NRrec > Ø then do
          call FillTab
          if rc = 8 then do
             call Free
             curØØ   = '@db2db'
             @db2msg =
             return
             end
          call SortEXL
          Call Wrrecjob
          "ispexec edit dataset('"outdsØ"')"
          call Free
          return
          end
       else do
          @db2msg = 'No record available for Database/Tablespace
selected'
          curØØ   = '@db2db'
          call Free
          return
          end
       end
    else do
       curØØ   = okunlo
       @db2msg = translate(NRrec,' ','_')
       call Free
       end
    return
   /*---  Routine fill up ISPF/TAB             ---*/
 FillTab:
   /*---  Read extracted Records               ---*/
    jobw = sysrecØØ
    "alloc da('"outdsrec"') f("jobw") shr reuse"
    xx=outtrap(trpreadØ1.)
```

```
   "execio * diskr sysrec00 (stem sysrec00. finis"
 xx=outtrap(off)
 if rc > 0 then do
    do #a = 1 to trpread01.0
       say trpread01.#a
    end
    say '' ; say ''
    say '>>>>>>>>'
    say '>>>>>>>>   Error reading file "'outdsrec'"    '
    say '>>>>>>>>   RC='rc'. Verify.                   '
    say '>>>>>>>>'
    say '' ; say '' ; call Free ; exit
    end
/*---  TBcreate TDISREC0                    ---*/
 address ispexec
 "tbcreate tdisrec0 names($tsname $ictype $opertyp $timesta)
  nowrite replace"
 DO #d = 1 to sysrec00.0
    $tsname  = strip(substr(sysrec00.#d,1,8))
    $ictype  = substr(sysrec00.#d,9,1)
    select
       when $ictype = 'F' then
          $opertyp  = 'Full Copy yes               '
       when $ictype = 'I' then
          $opertyp  = 'Full Copy no                '
       when $ictype = 'P' then
          $opertyp  = 'Recover TOcopy/Recover TOrba '
       when $ictype = 'Q' then
          $opertyp  = 'Quiesce                     '
       when $ictype = 'R' then
          $opertyp  = 'Load Replace log(yes)       '
       when $ictype = 'S' then
          $opertyp  = 'Load Replace log(no)        '
       when $ictype = 'W' then
          $opertyp  = 'Reorg log(no)               '
       when $ictype = 'X' then
          $opertyp  = 'Reorg log(yes)              '
       when $ictype = 'Y' then
          $opertyp  = 'Load log(no)                '
       when $ictype = 'Z' then
          $opertyp  = 'Load log(yes)               '
       when $ictype = 'T' then
          $opertyp  = 'Term utility command        '
       otherwise
          @db2msg= ' Error Ictype. End elaboration '
          exit
          end
     $timesta = strip(substr(sysrec00.#d,10,26))
    "tbmod tdisrec0"
```

```
      end  /* END DO #d = 1 to sysrec00.0 */
    "tbsort tdisrec0 fields($tsname,c,a,$timesta,c,d)"
    "tbtop  tdisrec0"
    "tbdispl tdisrec0 panel(@db2p011) cursor("cur00")"
    return
   /*---  Routine sort Execute dup record       ---*/
 SortEXL  :
    xx=outtrap(trpdummy.)
       "free fi(sortin)" ; "free fi(sortout)"
       "free fi(sysout)" ; "free fi(sysin)"
    xx=outtrap(off)
    "alloc da('"outdsrec"') f("sortin") shr reuse"
    "alloc da('"outdsrec"') f("sortout") shr reuse"
   /*---  File SYSOUT allocation for Sort      ---*/
    "alloc dummy f(SYSOUT)"
   /*---  File SYSIN allocation for Sort       ---*/
    jobw = sysin
    "alloc da('"outdsin"') f("sysin") shr reuse"
sk.1=' SORT FIELDS=(1,8,A),FORMAT=CH                    '
sk.2=' SUM FIELDS=NONE                                  '
sk.3=' END                                              '
    sk.0 = 3
    call WriteRec
    xx=outtrap(trp00.)
       "ispexec select pgm(SORT) "
       if rc > 0 then do
          say '' ; say '>>>>>>'
          say '>>>>>> SORT RC = 'rc
          say '>>>>>> Call Support Staff !!!!'
          say '>>>>>>' ; say '' ; call free ; exit
          end
    xx=outtrap(off)
    return
   /*---  Routine write Modify job             ---*/
 Wrrecjob :
    "alloc da('"outdsrec"') f("sysrec00") shr reuse"
    xx=outtrap(trpread01.)
       "execio * diskr sysrec00 (stem sysrec00. finis"
    xx=outtrap(off)
    if rc > 0 then do
       do #a = 1 to trpread01.0
          say trpread01.#a
       end
       say '' ; say ''
       say '>>>>>>>>'
       say '>>>>>>>>  Error reading file "'outdsrec'"    '
       say '>>>>>>>>  RC='rc'. Verify.                   '
       say '>>>>>>>>' ; say '' ; say '' ; call Free ; exit
       end
```

```
    outds0= $hiwork'.'@db2subs'.'$user'.#DB2TL0B.JOBREC'
    prmalloc = @db2subs' 'outds0' 0 1,1 f,b 80 2720 fijob no'
    call @db2all0 prmalloc
    if word(result,1) = 99 then exit
sk.1='//'userid()'Y  JOB ('$accn'),''DB2-Gestione'',CLASS='$class',   '
sk.2='//     MSGCLASS='$msgcla',REGION='$region',MSGLEVEL=('$msglvl'),'
sk.3='//     NOTIFY='$notif
sk.4='/*JOBPARM BYTES=999999,LINES=999999                             '
sk.5='//* ------------------------------------------------- *       '
sk.6='//* -------        Modify Tablespace           -------- *       '
sk.7='//* ------------------------------------------------- *       '
sk.8='//DB2PROC   JCLLIB ORDER=('$proclib')                           '
sk.9='//JOBLIB    DD  DISP=SHR,DSN='$dsnload
sk.10='//* ------------------------------------------------- *      '
sk.11='//MODIFY00 EXEC '$dsnproc',PARM='''@db2subs',MODY'||@db2db''''
sk.12='//SYSIN    DD  *                                               '
    #sx = 12
    DO #d = 1 to sysrec00.0
       #sx = #sx + 1
       $tsname  = substr(sysrec00.#d,1,8)
sk.#sx='MODIFY TABLESPACE '@db2db'.'$tsname 'DELETE AGE(*)            '
       end
    #sx = #sx + 1
sk.#sx='//* ------------------------------------------------- *     '
    #sx = #sx + 1
sk.#sx='//REXX00   EXEC   DB2REXX1                                    '
    #sx = #sx + 1
sk.#sx='//REXX00.SYSTSIN  DD  *                                       '
    #sx = #sx + 1
sk.#sx='  ISPSTART CMD(@DB2FOR0 '@db2subs','@db2db')                  '
    #sx = #sx + 1
sk.#sx='//* ------------------------------------------------- *     '
    sk.0=#sx
    jobw = fijob
    call WriteRec
    return
   /*---  Write record routine                ---*/
 WriteRec :
    address tso "EXECIO * DISKW "jobw" (STEM sk. FINIS"
 ClearRec:
    DO f = 1 to sk.0
       sk.f = blk
       end
    return
   /*---  Free & delete work DataSet          ---*/
 Free0 :
    xx=outtrap(trpdummy.)
       address tso "delete '"outdsprt"'"  ; address tso "delete
'"outdsrec"'"
```

```
      "free fi(systsinp)" ; address tso "delete '"outdstsin"'"
      "free fi(sysprint)" ; "free fi(sysrecØØ)"
      "free fi(syspunch)" ; "free fi(sysin)"
      address tso "delete '"outdsin"'" ; "free fi(fijob)"
      address tso "delete '"outdsØ"'"
  /*---  Free dataset Sort                    ---*/
      "free fi(sortin)" ; "free fi(sortout)"
      "free fi(sysout)" ; address tso "delete '"outsysin"'"
   xx=outtrap(off)
   return
  /*---  Free work DataSet                    ---*/
Free  :
   xx=outtrap(trpdummy.)
      address tso
      "free fi(systsinp)" ; "free fi(sysprint)"
      "free fi(sysrecØØ)" ; "free fi(syspunch)"
      "free fi(sysin)"    ; "free fi(fijob)"
      "free fi(sortin)"   ; "free fi(sortout)" ; "free fi(sysout)"
   xx=outtrap(off)
   return
```

$DB2FOR0 REXX EXEC

```
/* REXX */
trace ?o
   /*---     RESET COPY PENDING STATE      ---*/
arg parmin
parm    = translate(parmin,' ',',')
nparm   = words(parm) ; subsys  = word(parm,1) ; datab = word(parm,2)
   /*---  Test input parameter                 ---*/
 if nparm < 2 then do
    say '' ; say ''
    say '>>>>>>>>'
    say '>>>>>>>>  Parameter string is incomplete !!!! ' parmin
    say '>>>>>>>>'
    say '' ; say '' ; exit
    end
   /*---  Parameters assignment                ---*/
 call @db2parØ subsys
 if word(result,1) = 99 then exit
 $lpar    =word(result,1) ;$accn   =word(result,2) ;$class
=word(result,3)
 $msgcla  =word(result,4) ;$region =word(result,5) ;$msglvl
=word(result,6)
 $notif   =word(result,7) ;$user   =word(result,8) ;$unitda
=word(result,9)
 $unitta  =word(result,1Ø);$esunit =word(result,11);$prt
=word(result,12)
```

```
 $hiwork  =word(result,13);$db2ver =word(result,14);$ctsubs
=word(result,15)
 $librexx =word(result,16);$parmlib=word(result,17);$proclib
=word(result,18)
 $jcllib  =word(result,19);$report =word(result,2Ø);$libexec
=word(result,21)
 $isptenu =word(result,22);$isppenu=word(result,23);$ispmenu
=word(result,24)
 $ispslib =word(result,25);$plilink=word(result,26);$sibmlnk
=word(result,27)
 $sortlib =word(result,28);$runlib =word(result,29);$dsnload
=word(result,3Ø)
 $tep2pgm =word(result,31);$tep2pln=word(result,32);$unlopgm
=word(result,33)
 $unlopln =word(result,34);$dunlopg=word(result,35);$dunlopl
=word(result,36)
 $dsnproc =word(result,37)
   /*---  Work areas initialization            ---*/
 blk =  ; #r  = 1
   /*---  SYSIN file allocation                ---*/
 "alloc dummy f(sysin)"
   /*---  SYSTSINP file allocation             ---*/
 say ''
 outds1= $hiwork'.'subsys'.SYSTSINP'
 prmalloc = subsys' 'outds1' Ø 5,1 f,b 8Ø 2792Ø systsinp si'
 call @db2allØ prmalloc
 if word(result,1) = 99 then exit
sk.1='DSN SYSTEM('subsys')                                  '
sk.2='-DIS DB('datab') SPACENAM(*) RESTRICT LIMIT(999)      '
sk.3='END                                                   '
 sk.Ø=3;
 "execio * diskw systsinp (stem sk. finis"
   /*---  SYSTSPRT file allocation             ---*/
 outds2= $hiwork'.'subsys'.SYSTSPRT'
 prmalloc = subsys' 'outds2' Ø 45,15 f,b 8Ø 2792Ø systspri si'
 call @db2allØ prmalloc
 if word(result,1) = 99 then exit
 tipcom = DISPLAY
 Call Rundb2
 analisi = substr(trpØ3.1,1,8)
 select
    when analisi = 'DSNE1ØØI' then do
   /*---  DB2 subsystem not active             ---*/
       subs = center(subsys,1Ø)
       say '>>>>>>' ; say '>>>>>>'
       say '>>>>>> DB2 subsystem --->'subs'<--- is not active'
       say '>>>>>>' ; say '>>>>>>'
       say '' ; say '' ; call free ; exit
       end
```

```
 when analisi = 'DSNT5ØØI' then do
/*---  Unavailable DB2 resource             ---*/
    say '>>>>>> Display K.O. Verify RC.  '
    say ''
    do #a = 1 to trpØ3.Ø
       say trpØ3.#a
       end
    say '' ; call free ; exit
    end
 when analisi = 'DSNT3Ø1I' then do
/*---  DataBase incorrect                   ---*/
    say '>>>>>> DataBase not defined         '
    say ''
    do #a = 1 to trpØ3.Ø
       say trpØ3.#a
       end
    say '' ; call free ; exit
    end
 when analisi = 'DSNT36ØI' then do
/*---  Display OK                           ---*/
    jobw = systspri
    "alloc da('"outds2"') f("jobw") mod reuse"
    "execio * diskw systspri (stem trpØ3. finis"
/*---  Macro elab. out display              ---*/
    xx=OUTTRAP(trpØ4.)
       "ispexec edit dataset('"outds2"') macro(@mdb2Ø31)"
    xx=OUTTRAP(OFF)
    if rc > Ø then do
       do #a = 1 to trpØ4.Ø
          say trpØ4.#a
          end
       exit
       end
/*---  Test output macro                    ---*/
    xx=outtrap(trpØ5.)
       "execio * diskr systspri (stem systspri. finis"
       "free fi(systspri)"
    xx=outtrap(off)
    if rc > Ø then do
       do #a = 1 to trpØ5.Ø
          say trpØ5.#a
          end
       say '' ; say ''
       say '>>>>>>>>'
       say '>>>>>>>>   Error reading file "'outds2'"    '
       say '>>>>>>>>   RC='rc'. Verify.                 '
       say '>>>>>>>>' ; say '' ; say '' ; exit
       end
    no_estr = substr(systspri.2,1,25)
```

```
       if no_estr = '******** NO SPACES FOUND ' then do
          do #a = 1 to trp03.0
             say trp03.#a
             end
          say '' ; call free ; exit
          end
       else do
          do #a = 1 to trp03.0
             say trp03.#a
             end
           say ''
          jobw = systsinp
          "alloc da('"outds1"') f("jobw") shr reuse"
sk.#r='DSN SYSTEM('subsys')                                    '
          do #h = 2 to systspri.0
             #r = #r + 1
             sp_estr  = word(systspri.#r,1)
sk.#r='-START DB('datab') SPACE('sp_estr') ACCESS(FORCE) '
             end
          #r = #r + 1
sk.#r='END                                                    '
          sk.0=#r
          call WriteRec
          tipcom = START
          Call Rundb2
          do #a = 1 to trp03.0
             say trp03.#a
             end
   /*---  Last Display                          ---*/
sk.1='DSN SYSTEM('subsys')                                    '
sk.2='-DIS DB('datab') SPACENAM(*) RESTRICT LIMIT(999)    '
sk.3='END                                                     '
          sk.0=3
          call WriteRec
          tipcom = DISPLAY
          Call Rundb2
          do #a = 1 to trp03.0
             say trp03.#a
             end
          say '' ; call free
          end
       end
    otherwise
       say ''
       say '>>>>>> Unpredictable error. End elaboration       '
       say ''
       do #a = 1 to trp03.0
          say trp03.#a
          end
```

```
      say '' ; exit
   end
exit
  /*---  DB2 command Routine                  ---*/
Rundb2 :
   wrk = center(tipcom,8)
   say '                                                          '
   say ' ********************************************************** '
   say '*                                              'time()'     *'
   say '*       Command     'wrk  'is running                      *'
   say '*                                                         *'
   say '*                         wait please ........            *'
   say '*                                                         *'
   say ' ********************************************************** '
   say ''
   xx=outtrap(trpØ3.)
     address tso "ex '"outds1"'"
   xx=outtrap(off)
   return
  /*---  Write routine                        ---*/
WriteRec :
   "EXECIO * DISKW "jobw" (STEM sk. FINIS"
Pulisci:
   DO #f = 1 to sk.Ø
      sk.#f = blk
      end
   return
  /*---  Free work dataset                    ---*/
Free :
   xx=outtrap(trpdummy.)
      "free fi(sysin)" ; address tso "delete '"outds1"'"
      address tso "delete '"outds2"'"
   xx=outtrap(off)
   return
```

$DB2P000 PANEL

```
)ATTR FORMAT(MIX)
 >  type(text)    intens(&acc) color(white)
 «  type(text)    intens(high) color(yellow)
 %  type(text)    intens(high) color(white)  skip(on)
 +  type(text)    intens(low)  color(green)  skip(on)
 *  type(output)  intens(high) color(yellow) caps(off)
 #  type(output)  intens(low)  color(green)  skip(on)
 $  type(input)   intens(low)  color(red)    pad(_)
 @  type(input)   intens(low)  color(red)
)BODY expand(\\)
+ &ZUSER   + \-\- %DB2 Tools Main Menu +-\-\  #Z       +#Z       +
```

```
%COMMAND ===>$ZCMD
+
+*Z
+
    %DB2 Subsystem   ==>$Z   +             >DB2 Version         ==>#Z +

 +\-\   +
+
+
+
+                              «A+- Recover Tablespace TOCOPY
+                              «B+- SYSCOPY managment
+
+
+
+
+              %Select option+$Z+
+
+
+
+
+
+
+                                                  ---- %PF1+Help %PF3+End
----
)INIT
 .HELP  = @DB2H000
 .ZVARS = '@db2data,@db2tim,@db2msg,@db2subs,@db2ver,@db2opt'

)PROC
 &@db2data = '&zday/&zmonth/&zyear &ztime'
 ver(&@db2subs,nonblank)
 ver(&@db2opt,list,A,B)
)END
```

$DB2P010 PANEL

```
)ATTR FORMAT(MIX)
 %  type(text)    intens(high) color(&txhigh) skip(on)
 +  type(text)    intens(low)  color(green)   skip(on)
 ?  type(text)    intens(high) color(red)     skip(on)
 <  type(text)    intens(high) color(yellow)  skip(on) hilite(blink)
 *  type(output)  intens(high) color(yellow)  caps(off)
 #  type(output)  intens(low)  color(green)   skip(on)
 $  type(input)   intens(low)  color(red)     pad(_)
)BODY expand(\\)
+ &ZUSER + \-\- %DB2 SYSCOPY Managment +-\-\  #Z       +#Z       +
%COMMAND ===>$ZCMD
```

```
+
+*Z
+
    %DB2 Subsystem   ==>#Z   +             %DB2 Version         ==>#Z +

   ------------------------------------------------------------------------
    %Database      ==>$Z       +         %Tablespace          ==>$Z    +

  +\-\- %Variables Description +-\-\

   +%Database          +- DataBase to be selected

   +%Tablespace name   +- Tablespace to select to display Recovery info









+                                        ---- %PF1+Help %PF3+End ----

)INIT
 .HELP  =  @db2hØ17
 .ZVARS = '@db2data,@db2tim,@db2msg,@db2subs,@db2ver, +
           @db2db,@db2tbsp'
)PROC
 &@DB2DATA = '&ZDAY/&ZMONTH/&ZYEAR &ZTIME'
 ver(&@db2db,nonblank)
 ver(&@db2tbsp,nonblank)
)END
```

$DB2P011 PANEL

```
)ATTR FORMAT(MIX)
 %  type(text)    intens(high) color(&txhigh) skip(on)
 +  type(text)    intens(low)  color(green)   skip(on)
 ?  type(text)    intens(high) color(red)     skip(on)
 *  type(output)  intens(high) color(yellow)  caps(off)
 #  type(output)  intens(high) color(green)   skip(on)
 <  type(output)  intens(high) color(white)   skip(on)
 $  type(input)   intens(low)  color(red)     pad(_)
)BODY expand(\\)
+ &ZUSER + \-\- %DB2 SYSCOPY Managment +-\-\  #Z       +#Z       +
%COMMAND ===>$ZCMD
+
```

```
+*Z
+
    %DB2 Subsystem   ==>#Z   +             %DB2 Version         ==>#Z +
+                                       ---- %PF1+Help %PF3+End ----
   ------------------------------------------------------------------------
    %Database      ==>#Z     +            %Tablespace          ==>#Z  +

   ------------------------------------------------------------------------
%   TSname    ICTYPE          Operation Type            yyyy-mm-dd hh:mm
+  --------   ------  ----------------------------   ------------------
)MODEL
  #Z            #Z    #Z                            +    #Z             +
)INIT
 .HELP  =  @db2hØ18
 .ZVARS = '@db2data,@db2tim,@db2msg,@db2subs,@db2ver,                  +
           @db2db,@db2tbsp,                                            +
           $tsname,$ictype,$opertyp,$timesta'
)PROC
 &@DB2DATA = '&ZDAY/&ZMONTH/&ZYEAR &ZTIME'
)END
```

$DB2H017 HELP PANEL

```
)ATTR
>  type(text) intens(low) color(green)
?  type(text) intens(low) color(red)
%  type(text) intens(low) color(turq)
`  type(text) intens(low) color(yellow)
{  type(text)
$  type(text) intens(low)
#  type(nt)
}  type(et)
)BODY DEFAULT(<!*)EXPAND(°°)
>-°-°{ Managment Tool for DB2 environment Help >°-°-$
}3.B${DB2 SYSCOPY Managment$
$
$    >The function{"DB2 SYSCOPY Managment">allows you to
$    >display information for the Recovery of Tablespaces and run
$    >the MODIFY utility. MODIFY should be run regulary to clear
$    >SYSIBM.SYSCOPY and SYSIBM.SYSLGRNX. These tables may contain
$    >considerable amounts of outdated information, particularly
$    >SYSIBM.SYSLGRNX, which can become very large and take up
$    >considerable amounts of space. By deleting
$    >outdated information from these tables, you can help improve
$    >performance for processes that access data from these
$    >tables.The following fields are required :$
$
$    %Database      ==>?-------->     %Tablespace       ==>?-------->
```

```
$
$  >-{Database$    >Is the Database name used in the WHERE-condition for
$                  >the query on SYSIBM.SYSCOPY
$
$  >-{Tablespace$  >Is the name of Tablespace for which it is asked to
$                  >carry out the display of the information of Recovery
$                  >and eventually prepare the job of{MODIFY.>
$                  >The field Tablespace allows the caracter &varst
$
$                                          {ENTER>to continue{PF3>End>
$
)INIT
&varst = '* and %'
)PROC
  &zcont = @db2hØ18
)END
```

$DB2H018 HELP PANEL

```
)ATTR
>  type(text) intens(low) color(green)
?  type(text) intens(low) color(red)
`  type(text) intens(low) color(yellow)
{  type(text)
$  type(text) intens(low)
#  type(nt)
}  type(et)
)BODY DEFAULT(<!*)EXPAND(°°)
>-°-°{ Managment Tool for DB2 environment Help >°-°-$
$
$    >After the selection of the{Database>and the{Tablespace>the rexx
$    >show the display of the Recovery information available.
$
$     ------------------------------------------------------------------
$      Database      ==>{YRDFIST>          $Tablespace          ==>{YR%>
$     ------------------------------------------------------------------
$     TSname     ICTYPE        Operation Type           yyyy-mm-dd hh:mm
$     --------   ------  --------------------------   -----------------
$     YRSFISTØ      Y     Load log(no)                  2ØØØ-Ø7-14-11.59
$     YRSFIST1      Y     Load log(no)                  2ØØØ-Ø7-15-11.59
$     YRSFIST2      Z     Load log(yes)                 2ØØØ-11-23-Ø9.23
$     YRSFISTØ      X     Reorg log(yes)                2ØØØ-11-24-14.25
$     YRSFISTØ      R     Load Replace log(yes)         2ØØØ-11-25-16.Ø2
$     YRSFISTØ      Y     Load log(no)                  2ØØØ-12-26-11.59
$
$
$    >Press{ENTER>to create the{MODIFY>job and edit it
$
```

```
$
$                                                ,{ENTER>to continue{PF3>End>
$
$
)INIT
)PROC
  &zcont = @db2h000
)END
```

$MDB2031 MACRO

```
/* REXX */
trace ?o
/*-----   Used in @DB2FOR0 Rexx            -----*/
isredit macro
isredit exclude all
isredit find '''DATABASE = '''
isredit find '''NO SPACES FOUND'''
isredit find ''',COPY''' 22   all
isredit find ''',CHKP''' 22  all
isredit find ''',RECP''' 22  all
isredit delete x all
isredit save
isredit end
```

Giuseppe Rendano
DB2 Systems Programmer (Italy)

Copying dictionary pages across subsystems

For compressed tablespaces, one of the DBA's functions is to maintain dictionary data pages. These dictionary pages are built when REORG or LOAD REPLACE utilities are run for the tablespace with a number of rows sufficient to build a good dictionary with a very high compression ratio. Once these dictionary pages are built, any subsequent updates (inserts/updates) to the table compress the updated rows, resulting in considerable savings in disk storage.

Often, DBAs will need to migrate the dictionary pages across the subsystems without migrating the data. This migration is also needed to build dictionary pages when a table (with compressed tablespace) is created for the first time in production and data is not available to build the dictionary. For a big application with many large compressed tablespaces having multiple partitions, this requirement is very challenging because migrating millions of rows for each partition, just to build dictionary pages, is not an easy task anyway. Furthermore, if dictionary pages are not present or they do not have a good compression ratio, subsequent inserts to the table will not create compressed rows – resulting in space problems.

In this article I have suggested a method that can be used to transfer dictionary pages for a compressed tablespace from a development subsystem to the production subsystem without any need for data migration. Here the target tablespace is in the production subsystem and the source tablespace is in the development subsystem. The suggested steps are as follows:

- Unload data from the source table in development and keep it in a safe place.
- Run LOAD REPLACE with the KEEPDICTIONARY option and DUMMY input data on the existing source table, which has compressed data. This will empty the tablespace data pages, retaining only dictionary pages. These dictionary pages will be migrated in subsequent steps.
- Take a full imagecopy of this tablespace (with the DSNUM ALL option).

- Create the target compressed tablespace and the target table in production.
- Note down DBIDs and PSIDs for the source and target tablespaces. Also, note down the OBIDs of the source and target tables.
- STOP the target tablespace in production.
- Run DSN1COPY with the imagecopy (taken above) dataset as input and target the tablespace underlying the VSAM dataset as output. For a partitioned tablespace, this output VSAM dataset will correspond to the first partition (A001) of the target tablespace. This DSN1COPY will also have OBIDXLAT and RESET options to translate DBIDs, PSIDs, and OBIDs.
- START the target tablespace in *Utility* mode.
- Run LOAD REPLACE with the KEEPDICTIONARY option and DUMMY input data on the target table. This step will change RBAs/LRSNs, SSIDs, VCATs, etc in the target tablespace pages (header and dictionary), making them consistent with the production subsystem.
- START the target tablespace in read/write mode.

For example, consider a table, SALES.TPRODUCT, in tablespace DSALES.SPRODUCT. SPRODUCT is a partitioned tablespaces with three partitions. In the development subsystem, table TPRODUCT has enough rows to build dictionary pages and the REORG utility was run on this tablespace to build a dictionary. To move this table into production, the tablespace DSALES.SPRODUCT has been created in the production subsystem and table SALES.TPRODUCT has been created in this tablespace. Tablespace SPRODUCT has been created with COMPRESS YES for all its three partitions.

To migrate dictionary pages for this tablespace do the following. Unload data from TPRODUCT table in development and save it. Run LOAD REPLACE on table TPRODUCT with dummy input and the KEEPDICTIONARY option in development. This will make the whole table empty without disturbing the dictionary pages. Run imagecopy on tablespace SPRODUCT with the DSNUM ALL option

in development and create the imagecopy dataset DSALES .SPRODUCT.IMAGCOPY. This imagecopy dataset has a *Header* page, a *Spacemap* page, and 16 *Dictionary* pages for each of the three partitions. The total number of pages copied is 54. Note down the DBID and PSID of the tablespace SPRODUCT in development and production, and the OBID of table TPRODUCT in development and production. Stop tablespace SPRODUCT in production. Run DSN1COPY with the OBIDXLAT and RSET option in production. SYSUT1 for this job is the imagecopy dataset DSALES .SPRODUCT.IMAGCOPY created in an earlier step. SYSUT2 for this job is PROD.DSNDBC.DSALES.SPRODUCT.I0001.A001, the underlying VSAM dataset for partition one of tablespace SPRODUCT in production. A sample job is shown below. Start tablespace SPRODUCT in *Utility* mode in production. Run LOAD REPLACE on table TPRODUCT with dummy input and the KEEPDICTIONARY option in production. This will reset all RBAs/LRSNs and subsystem-IDs in the tablespace pages. Start tablespace SPRODUCT in read/write mode in production.

Table TPRODUCT is now ready for all updates in production.

DSN1COPY JOB

```
//SALESJB1 JOB (P804,0000),'DSN1COPY -------',
//         CLASS=D,MSGCLASS=X,NOTIFY=USER1P
//COPY   EXEC PGM=DSN1COPY,
//            PARM='OBIDXLAT,NUMPARTS(3),FULLCOPY',COND=(4,LT)
//STEPLIB  DD DSN=PROD1.PROD.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=SHR,
//         DSN=DSALES.SPRODUCT.IMAGCOPY
//SYSUT2   DD DISP=OLD,AMP=('AMORG,BUFND=21'),
//         DSN=PROD.DSNDBC.DSALES.SPRODUCT.I0001.A001
//SYSXLAT   DD  *
0006,0006
0271,0198
0272,0199
/*
//
```

Sharad K Pande
Senior DBA
PriceWaterhouseCooper (USA)

DB2 routines administration – part 2

This month we conclude the article describing DB2 admistration routines.

```
Address DSNREXX "EXECSQL DECLARE C1 CURSOR FOR S1"
Address DSNREXX 'EXECSQL PREPARE S1 FROM :SQLSTMT'
Address DSNREXX "EXECSQL OPEN C1"
Address DSNREXX "EXECSQL FETCH C1 INTO :HVST, :HVFU"

address ispexec 'tbcreate "dlist" names(item)'

do while(sqlcode=Ø)
   item=hvfu
   address ispexec 'tbadd "dlist"'
   Address DSNREXX "EXECSQL FETCH C1 INTO :HVST, :HVFU"
end
code = sqlcode
Address DSNREXX "EXECSQL CLOSE C1"
item=';'
address ispexec 'tbadd "dlist"'
/* End Part IIb.   ************************************** */

Call Grant
end

if x=3 then do
SQLSTMT= "SELECT STRIP(SCHEMA), NAME,                                ",
"         CASE(TRIGTIME)                                             ",
"           WHEN('B') THEN 'BEFORE '                                 ",
"           WHEN('A') THEN 'AFTER '                                  ",
"         END CONCAT                                                 ",
"         CASE(TRIGEVENT)                                            ",
"           WHEN('I') THEN 'INSERT'                                  ",
"           WHEN('U') THEN 'UPDATE'                                  ",
"           WHEN('D') THEN 'DELETE'                                  ",
"         END,                                                       ",
"         STRIP(TBOWNER)||'.'||STRIP(TBNAME),                        ",
"         STRIP(REMARKS), STRIP(TEXT)                                ",
"FROM SYSIBM.SYSTRIGGERS                                             ",
"WHERE SCHEMA LIKE '"rsche"%'                                        ",
"  AND NAME   LIKE '"rname"%'                                        ",
"  WITH UR                                                           "

Address DSNREXX "EXECSQL DECLARE C1 CURSOR FOR S1"
Address DSNREXX 'EXECSQL PREPARE S1 FROM :SQLSTMT'
```

```
Address DSNREXX "EXECSQL OPEN C1"
Address DSNREXX,
 "EXECSQL FETCH C1 INTO :HVSC, :HVNA, :HVSP, :HVEN, :HVRE, :HVPK"

address ispexec,
        'tbcreate "rlist" names(HVSC,HVNA,HVSP,HVEN,HVRE,HVPK)'

do while(sqlcode=Ø)
   address ispexec 'tbadd "rlist"'
   Address DSNREXX,
    "EXECSQL FETCH C1 INTO :HVSC, :HVNA, :HVSP, :HVEN, :HVRE, :HVPK"
end
code = sqlcode
Address DSNREXX "EXECSQL CLOSE C1"
address ispexec 'tbtop "rlist"'
address ispexec 'tbdispl "rlist" panel(UDFM1)'
if rc=8 then do
   address ispexec 'tbend "rlist"'
   address ispexec "tbend "messdb""
   Call Create_messg
   Signal Top
end
if rc<8 & cmd='S' then nop
else do
   address ispexec 'tbend "rlist"'
   address ispexec "tbend "messdb""
   Call Create_messg
   Signal Sub
end
address ispexec 'tbend "rlist"'
Call Create_trigger
Call Grant
end

if code=Ø | code=1ØØ then nop
else do
   if code=1ØØ
   then message='Routine '||function||' not found.'
   else message='Error. Sqlcode = '||code
   address ispexec "tbend "messdb""
   Call Create_messg
   Call Error 'X'
end

messg = 'Building the Create Statements'
messg = time() || " " || messg
Call Send_messg
/* JCL Skeleton DB2 Routines              */
date=date()
```

```
time=time(c)
user=userid()
tempfile=userid()||'.UTIL.UDF.JCL'
address tso
"delete '"tempfile"'"
"free dsname('"tempfile"')"
"free ddname(ispfile)"
"free attrlist(formfile)"
"attrib formfile blksize(800) lrecl(80) recfm(f b) dsorg(ps)"
"alloc ddname(ispfile) dsname('"tempfile"')",
       "new using (formfile) unit(3390) space(1 1) cylinders"
address ispexec
"ftopen"
"ftincl UDFSP"
"ftclose"
zedsmsg = "JCL shown"
zedlmsg = "JCL DB2 Routines shown"
"setmsg msg(isrz001)"
"edit dataset('"tempfile"')"
/* address ispexec  "tbclose "messdb"" */
address ispexec 'tbend "dlist"'
exit
Grant:
 if x=1 | x=2 then do
 messg = "Select     sysroutineauth information"
 messg = time() || " " || messg
 Call Send_messg

 item='-- Grant statements'
 address ispexec 'tbadd "dlist"'

 /* Part III. Grant privilege                                  */
 if x=1
 then routine='SPECIFIC FUNCTION'
 else routine='PROCEDURE'
 SQLSTMT=,
 " SELECT 'GRANT EXECUTE ON "routine" '||STRIP(SCHEMA)||          ",
 "   '.'||STRIP(SPECIFICNAME), ' TO '||STRIP(GRANTEE)||           ",
 "        CASE                                                    ",
 "         WHEN(EXECUTEAUTH)='G'                                  ",
 "          THEN ' WITH GRANT OPTION;'                            ",
 "          ELSE ';'                                              ",
 "        END                                                     ",
 "FROM SYSIBM.SYSROUTINEAUTH                                     ",
 "WHERE SPECIFICNAME='"specname"'                                ",
 "  AND GRANTOR<>GRANTEE                                         ",
 "  AND ROUTINETYPE='"type"'                                     ",
 "WITH UR                                                        "
```

```
Address DSNREXX "EXECSQL DECLARE C1 CURSOR FOR S1"
Address DSNREXX 'EXECSQL PREPARE S1 FROM :SQLSTMT'
Address DSNREXX "EXECSQL OPEN C1"
Address DSNREXX "EXECSQL FETCH C1 INTO :HVF1, :HVF2"
if sqlcode=Ø then do
   item='COMMIT;'
   address ispexec 'tbadd "dlist"'
end
do while(sqlcode=Ø)
   item=hvf1
   address ispexec 'tbadd "dlist"'
   item='     '||hvf2
   address ispexec 'tbadd "dlist"'
   Address DSNREXX "EXECSQL FETCH C1 INTO :HVF1, :HVF2"
end
Address DSNREXX "EXECSQL CLOSE C1"
end

/*                                                          */
messg = "Select     syspackauth    information"
messg = time() || " " || messg
Call Send_messg

if x=3 then hvpk = hvna

SQLSTMT=,
" SELECT SUBSTR(AUTH,1,LENGTH(AUTH)-2), OPT                    ",
" FROM (                                                       ",
" SELECT 'GRANT '||                                            ",
"        CASE                                                  ",
"         WHEN(BINDAUTH)='Y' THEN 'BIND, ' ELSE ''             ",
"        END ||                                                ",
"        CASE                                                  ",
"         WHEN(COPYAUTH)='Y' THEN 'COPY, ' ELSE ''             ",
"        END ||                                                ",
"        CASE                                                  ",
"         WHEN(EXECUTEAUTH)='Y' THEN 'EXECUTE, ' ELSE ''       ",
"        END AUTH,                                             ",
"        ' ON PACKAGE '||                                      ",
"        STRIP(COLLID)||'.* TO '||STRIP(GRANTEE)||             ",
"        CASE                                                  ",
"         WHEN(EXECUTEAUTH)='G'                                ",
"          THEN ' WITH GRANT OPTION;'                          ",
"          ELSE ';'                                            ",
"        END OPT                                               ",
"FROM SYSIBM.SYSPACKAUTH                                       ",
"WHERE COLLID='"hvpk"'                                         ",
"  AND GRANTOR<>GRANTEE ) X                                    ",
"WITH UR                                                       "
```

```
 Address DSNREXX "EXECSQL DECLARE C1 CURSOR FOR S1"
 Address DSNREXX 'EXECSQL PREPARE S1 FROM :SQLSTMT'
 Address DSNREXX "EXECSQL OPEN C1"
 Address DSNREXX "EXECSQL FETCH C1 INTO :HVF1, :HVF2"
 if sqlcode=Ø then do
    item='COMMIT;'
    address ispexec 'tbadd "dlist"'
 end
 do while(sqlcode=Ø)
    item=hvf1
    address ispexec 'tbadd "dlist"'
    item='     '||hvf2
    address ispexec 'tbadd "dlist"'
    Address DSNREXX "EXECSQL FETCH C1 INTO :HVF1, :HVF2"
 end
 Address DSNREXX "EXECSQL CLOSE C1"

 address ispexec 'tbtop "dlist"'
 /* End Part III. ************************************* */
Return
Create_trigger:
address ispexec 'tbcreate "dlist" names(item)'
item=delword(hvpk,6)
address ispexec 'tbadd "dlist"'
text=subword(hvpk,6)
do i=1 to length(text)
   if substr(text,i,1)=',' then text=insert(' ',text,i)
end
item=''
do i=1 to words(text)
   item=item||word(text,i)||' '
   if length(item) > 4Ø then do
      address ispexec 'tbadd "dlist"'
      item=''
   end
end
if length(item) > Ø then address ispexec 'tbadd "dlist"'
item=' '
address ispexec 'tbadd "dlist"'
Return
Error:
 ARG cur_par
 cur=cur_par
 address ispexec "setmsg msg(udfØØ1)"
 signal top
Return
Create_messg:
 messg = "s"||userid()
```

```
  address ispexec "tbcreate "messdb" names(messg) write replace"
 Return
 Send_messg:
  address ispexec "tbadd " messdb
  address ispexec "control display lock "
  address ispexec "addpop row(13) column(6)"
  address ispexec "tbdispl "messdb" panel(udfmes)"
  address ispexec rempop
 Return
```

UDFM0 PANEL

```
)ATTR DEFAULT(%+_)
   [ TYPE (OUTPUT) INTENS(LOW)  COLOR(GREEN) CAPS(OFF)
   # TYPE (OUTPUT) INTENS(LOW)  COLOR(WHITE) CAPS(OFF)
   ] TYPE (TEXT)   INTENS(LOW)  COLOR(WHITE) CAPS(OFF)  HILITE(REVERSE)
   _ TYPE (INPUT)  INTENS(LOW)  COLOR(YELLOW) CAPS(ON)  HILITE(BLINK)
   | TYPE (OUTPUT) INTENS(LOW)  COLOR(GREEN)  CAPS(OFF)
   + TYPE (TEXT)   INTENS(LOW)  COLOR(GREEN)
   / TYPE (TEXT)   INTENS(LOW)  COLOR(TURQ)
   ~ TYPE (TEXT)   INTENS(HIGH) COLOR(TURQUOISE)
   @ TYPE (TEXT)   INTENS(HIGH) COLOR(RED)   CAPS(OFF)  HILITE(REVERSE)
)BODY WINDOW(41,15) EXPAND ($$)
+]                   + Date:|date       +
+] DB2 Routines      + Time:|time      +
+]                   + User: &zuser
/ ************************************
+
+  [row1                              +
+  [row2                              +
+  [row3                              +
+  [row4                              +
+
/ ************************************
+
+@==>+ _X+  #msg                       +
+
+] PF1 Help +                ] PF3 End +
)INIT
  &row1= '1 - User-Defined Function UDF'
  &row2= '2 - Stored Procedure'
  &row3= '3 - Triggers'
  &row4= 'X - Exit'
  IF (&X = 1,2,X)
      &msg = ''
  ELSE
      .ATTR (msg) = 'COLOR (RED)'
      &msg = 'Enter 1, 2, 3 or X.'
```

```
  IF (&X = 1)
      .ATTR (row1) = 'COLOR (YELLOW) CAPS(ON)'
  IF (&X = 2)
      .ATTR (row2) = 'COLOR (YELLOW) CAPS(ON)'
  IF (&X = 3)
      .ATTR (row3) = 'COLOR (YELLOW) CAPS(ON)'
)PROC
  IF (.PFKEY = PFØ3) &PF3 = EXIT
)END
```

UDFM1 PANEL

```
)Attr Default(%+_)
   | type(text)   intens(high) caps(on ) color(yellow)
   $ type(output) intens(high) caps(off) color(yellow) hilite(reverse)
   $ type(output) intens(high) caps(off) color(white) hilite(reverse)
   ? type(text)   intens(high) caps(on ) color(green) hilite(reverse)
   # type(text)   intens(high) caps(off) hilite(reverse)
   } type(text)   intens(high) caps(off) color(white)
   [ type( input) intens(high) caps(on ) just(left )
   { type( input) intens(high) caps(on ) just(left ) pad('_')
   ] type( input) intens(high) caps(on ) just(left ) pad('-')
   ^ type(output) intens(low ) caps(off) just(asis ) color(turquoise)
)Body  Expand(//)
%-/-/- $title                    +%-/-/-
%Command ===>_zcmd                                        / /%Scroll
===>_amt +
+SSID[db2 +
+-------------------------------------------------------------------------
--------
+Valid cmd:|S+Continue
}F3+->}End
+-------------------------------------------------------------------------
--------
#S#Schema  #Name              $title1            +$title2
+
  {rsche   {rname             {rspec              {rexte
+
)Model
]z^z       ^z                ^z                  ^z
+
)Init
  .ZVARS = '(cmd HVSC HVNA HVSP HVEN)'
  &amt = PAGE
  &cmd = ''
  if (&rsche ^= ' ')
      .attr (rsche) = 'pad(nulls)'
  if (&rname ^= ' ')
```

```
      .attr (rname) = 'pad(nulls)'
  if (&rspec ^= ' ')
      .attr (rspec) = 'pad(nulls)'
  if (&rexte ^= ' ')
      .attr (rexte) = 'pad(nulls)'
)Reinit
)Proc
  VPUT (db2, rsche, rname, rspec, rexte) PROFILE
)End
```

UDFMES PANEL

```
)ATTR DEFAULT(%+_)
| TYPE (TEXT)   INTENS(LOW)  COLOR(WHITE)
@ TYPE (TEXT)   INTENS(HIGH) COLOR(RED)   CAPS(OFF) HILITE(REVERSE)
| TYPE (INPUT)  INTENS(NON)  COLOR(GREEN) CAPS(ON)  JUST(LEFT)
# TYPE (OUTPUT) INTENS(LOW)  COLOR(GREEN) CAPS(OFF)
)BODY DEFAULT(%~\) WINDOW(6Ø,8)
|ZCMD +            @ Message display |AMT  |
|-------------------------------------------------------------
)MODEL CLEAR(MESSG)
#Z                                        +
)INIT
     .ZVARS = '(MESSG)'
)REINIT
)PROC
  IF (.PFKEY = PFØ3) &PF3 = EXIT
  IF (&ZCMD=END)
     &COMMAND = CANCEL
)END
```

UDF00 MESSAGE

```
UDFØØ1           .ALARM = YES  .WINDOW=NORESP .ALARM = YES
'&message'
```

UDFSP JCL SKELETON

```
)TBA 72
)CM ------------------------------------------------------------------
)CM JCL Skeleton: DB2 Routines - UDF and SP                          --
)CM ------------------------------------------------------------------
//&user.X JOB (12ØØ-12Ø5-ØØ),'&option',
//             NOTIFY=&user,REGION=4M,
//             CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//RUNSQL  EXEC PGM=IKJEFTØ1
```

```
//STEPLIB DD DISP=SHR,DSN=DSN61Ø.SDSNLOAD
//        DD DISP=SHR,DSN=CEE.SCEERUN
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(&db2)
 RUN PROGRAM(DSNTEP2) PLAN(DSNTEP61) -
     LIB('DSN61Ø.RUNLIB.LOAD.DSNN')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *
 SET CURRENT SQLID = '&user' ;
)SEL &x EQ 1
)BLANK 1
 --DROP SPECIFIC FUNCTION &hvsc..&hvsp RESTRICT;
 --COMMIT;
)BLANK 1
)ENDSEL
)SEL &x EQ 2
)BLANK 1
 --DROP PROCEDURE &hvsc..&hvsp RESTRICT;
 --COMMIT;
)BLANK 1
)ENDSEL
)SEL &x EQ 3
)BLANK 1
 --DROP TRIGGER &hvsc..&hvna RESTRICT;
 --COMMIT;
)BLANK 1
)ENDSEL
)DOT "DLIST"
 &item
)ENDDOT
)BLANK 1
)SEL &x EQ 1
 COMMENT ON SPECIFIC FUNCTION &hvsc..&hvsp IS
 '&hvre';
)ENDSEL
)SEL &x EQ 2
 COMMENT ON PROCEDURE &hvsc..&hvsp IS
 '&hvre';
)ENDSEL
)SEL &x EQ 3
 COMMENT ON TRIGGER &hvsc..&hvna IS
 '&hvre';
)ENDSEL
```

Bernard Zver
DBA (Slovenia)

An introduction to DB2 UDB text extenders

SUMMARY

Being able to search large amounts of data for particular words or phrases is becoming more and more important to businesses, particularly those that are Web based. DB2 has addressed this issue with two offerings – Text Information Extender (TIE) and the Net Search Extender (NSE).

INTRODUCTION

This article compares the DB2 UDB 7.2 EXTENDER products. It discusses the differences between the Text Information Extender (TIE) offering and the Net Search Extender (NSE) offering (Text Extender (TE) is not discussed).

HISTORY

Text extenders have been around since about 1995, with TE being the first offering, followed by NSE, and finally TIE (which was incorporated into UDB 7.2). TIE will replace TE as soon as it covers all the functionality and platforms of TE.

WHY USE TEXT EXTENDERS?

The advantage of using text extenders over standard SQL (eg LIKE) is basically two fold – namely speed of search and improved search criteria specification. Both TIE and NSE handle the speed and search issues differently, and your choice depends on your set-up and what type of query you want to execute.

Text extenders come into their own when you are searching large amounts of data (many GB's worth), you need the results quickly, and you are not sure of the exact string you are searching for! They were developed primarily for Web-based applications requiring a search function.

KEY TEXT EXTENDER FEATURES

Figure 1 lists all the features of the two text extenders. A '✓' means the function is supported, and an 'X' means it isn't.

In summary, they allow you to do:

- Fuzzy searches – if you are unsure as to how the word is spelt.

Function	*NSE*	*TIE*
Thesaurus support	X	✓
Section support	✓	✓
Boolean support	✓	✓
'In same sentence as' support	✓	✓
'In same paragraph as' support	X	✓
'Fuzzy form of' support	✓	✓
'Stemmed form of' support	✓	✓
'Precise form of' support	X	✓
Masking character support (single and multiple)	✓	✓
NUMBEROFMATCHES support	X	✓
SCORE support	X	✓
Searching for terms in a fixed sequence	X	✓
Free text search support	X	✓
Hybrid search support	X	✓
Indexes can be stored in main memory	✓	X
Supported text index types: CHAR, VARCHAR, LONG VARCHAR	✓	✓
Supported: GRAPHIC,VARGRAPHIC,LONG VARGRAPHIC	X	✓
Supported: BLOB,DATALINK,DBCLOB,CLOB	X	✓
Is the data type DATALINK supported?	X	✓
Can indexes be reorged?	X	✓
Can you check if an index is being updated?	X	✓
Can locking services be changed?	X	✓
Is it supported on an EE system?	✓	✓
Is it supported on an EEE system?	✓	X
Can it handle multiple column primary keys?	✓	X
Linguistic support?	X	X
Can you index data stored on files referenced using the DB2 datalink feature?	X	✓

Figure 1: Differences between TIE and NSE functionality

- Masking of individual or multiple characters in a search string.
- In the same sentence/paragraph – allows you to search for two or more words, which then have to be in the same sentence/ paragraph.
- Free text searches – allows users to put in free text and then the text extenders will search on the nouns in the search string.
- Thesaurus searches – you can set up your own list of synonyms/ relationships, based on your business needs.

As you can see, there are many excellent features which make DB2 text extenders ideal for searching on the Web.

Platforms supported are shown in Figure 2.

Platform:	*TIE*	*NSE*
AIX	✓	✓
Sun Solaris	✓	✓
HP	X	✓
Windows (NT/2000)	✓	✓
OS/390	✓	✓

Figure 2: Platforms supported

MANUALS

IBM DB2 Universal Database Text Information Extender Administration and User's Guide Version 7.2, SH12-6732-00.

DB2 Universal Database DB2 Net Search Extender Administration and Programming Version 7.1.1, SC27-0818-01.

Both of these manuals are available when the products are installed.

FUNDAMENTAL CHOICES

There are operational limitations to TIE and NSE, and these are shown below. Before deciding on functionality, you need to make sure that your environment supports the text extender you want to use.

You can't use TIE if:

- You are running on an Extended Enterprise Edition system.
- The primary keys of the tables you wish to index consist of multiple columns.

FUNCTIONAL CHOICES

Below are some of the key factors which will affect your choice of extender:

- NSE
  - Runs in a single stored procedure and the query cannot be combined with other SQL predicates.
  - Can only perform text searches.
- TIE
  - Can be incorporated into standard SQL statements (thus using predicates).
  - Can perform text and non-text searches.
  - Can perform searches on files referenced using the DB2 datalink feature.

You may have to adapt your application to fit in with the functionality of each text extender.

AND FINALLY…

I hope I have given you a feel for what each text extender can do. TIE is being constantly improved in terms of functionality and to work on all platforms (like HP). Check out the DB2 Web page for announcements. Give it a go, and see what it can do for you!

And don't forget that TIE is free!! (it comes on its own CD).

C Leonard
Freelance Consultant (UK)

BMC Software has announced that it will credit its customers the costs of IBM's DB2 utilities. With the release of DB2 Version 7, IBM elected to charge customers for utilities that were previously free.

BMC's rival products cover performance management, recovery management, and database administration.

For further information contact:
BMC Software, 2101 City West Blvd, Houston, TX 77042-2827, USA.
Tel: (713) 918 8800.
URL: http://www.bmc.com/solutions/database.

* * *

BakBone Software has launched a suite of NetVault Application Plug-in Modules (APMs) for DB2 and Lotus Notes, covering all major Unix, Linux, and Windows platforms and enabling the back-up and restore of files while the application is on-line and active.

The new APMs offer multiple back-up modes and drill down to individual mailboxes, databases, or tablespaces.

The DB2 modules include on-line parallel back-up and restore, on-line back-up of entire database, tablespace, and archive logs, plus remote management and centralized on-line incremental back-ups.

It can also direct a database recovery to an alternative location, and support AIX, Solaris, Linux, Windows NT/2000, and other platforms.

For further information contact:
BakBone, 10145 Pacific Heights Blvd, Suite 900, San Diego, CA 92121, USA.
Tel: (877) 939 2663.
URL: http://www.bakbone.com/products/netvault/apm.asp.

* * *

Princeton Softech has announced Relation ToolsT 5.0, a testing toolset for extracting, migrating, editing, and comparing related sets of DB2 and legacy data.

This new release includes Move for LegacyT, which integrates VSAM and sequential data with DB2 sources into a consistent test environment.

For further information contact:
Princeton Softech, 111 Campus Drive, Princeton, NJ 08540-6400 USA.
Tel: (609) 627 5500.
URL: http://www.princetonsoftech.com/products/index.htm.

* * *

IBM has announced new and deeper integration between Lotus Domino, WebSphere Application Server, and DB2 Universal Database in the next version of Domino.

For enterprise applications with high-end data requirements, the new Domino with Lotus Enterprise Integrator can use DB2 as the primary data source, allowing it to operate in a multi-tier application architecture to extend the scalability of Domino applications.

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