64 DB2

February 1998

In this issue

update

DB2 Update

Published by

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

North American office

Xephon
1301 West Highway 407, Suite 201-450
Lewisville, TX 75067, USA
Telephone: 940 455 7050

Australian office

Xephon/RSM
PO Box 6258, Halifax Street
Adelaide, SA 5000
Australia
Telephone: 08 223 1391

Contributions

If you have anything original to say about DB2, or any interesting experience to recount, why not spend an hour or two putting it on paper? The article need not be very long – two or three paragraphs could be sufficient. Not only will you be actively helping the free exchange of information, which benefits all DB2 users, but you will also gain professional recognition for your expertise, and the expertise of your colleagues, as well as some material reward in the form of a publication fee – we pay at the rate of £170 ($250) per 1000 words for all original material published in *DB2 Update*. If you would like to know a bit more before starting on an article, write to us at one of the above addresses, and we'll send you full details, without any obligation on your part.

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 £235.00 in the UK; $350.00 in the USA and Canada; £241.00 in Europe; £247.00 in Australasia and Japan; and £245.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1994 issue, are available separately to subscribers for £20.00 ($30.00) each including postage.

DB2 Update on-line

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

Printed in England.

DB2 command interface

INTRODUCTION

The DB2CT service (a REXX procedure) provides a DB2 command interface that enables you to:

- Display, start, or stop databases and tablespaces:
  - The DB2 DISPLAY DATABASE command displays information about the status of DB2 databases, tablespaces, and index spaces, and also indicates whether a tablespace (partition) or index space is in a ‘pending’ state.
  - The DB2 STOP DATABASE command makes the specified databases or tablespaces unavailable for applications, and closes their datasets.
  - The DB2 START DATABASE command is typically used after a previous STOP DATABASE command, or after a tablespace or index has been placed in a deferred restart status by DB2. It makes the specified database available for use. Depending on the options you specify, the database-tablespace can be made available for RO, RW, UT, or FORCE processing.
- Display or cancel thread:
  - The DB2 command DISPLAY THREAD displays the current status information for DB2 threads. In this case, it displays only distributed threads.
  - The DB2 command CANCEL DDF THREAD cancels processing for threads that originate locally and access remote data, or originate remotely and access local data.
- Display or terminate utility:
  - The DB2 DISPLAY UTILITY command displays the status of utility jobs. A job can be active, stopped, or terminating.

  - The DB2 TERM UTILITY command terminates execution of a utility job step and releases all resources associated with the step.

- Start or stop DDF:
  - The START DDF command starts the Distributed Data Facility (DDF).
  - The STOP DDF command stops the Distributed Data Facility (DDF).

THE COMPONENTS OF DB2CT

The following are the components of DB2CT.

SYSPROC:

- DB2CT – driver procedure
- DISDB – command procedure for database-tablespace(s)
- DISTR – command procedure for DDF thread(s)
- DISUT – command procedure for utility
- DB2DDF – start, stop DDF procedure.

ISPPLIB:

- DB2CTP00 – main menu
- DISDBM00 – display database-tablespace panel
- DISDBM01 – display database-tablespace panel (additional information)
- DISTRM00 – display thread panel
- DISUTM00 – display utility panel
- DB2DDF00 – start, stop DDF panel.

ISPMLIB:

- DISDB00 – db2ct message.

START DB2CT PROCEDURE

The DB2CT procedure can be invoked by issuing the TSO DB2CT statement in ISPF (see Figure 1).

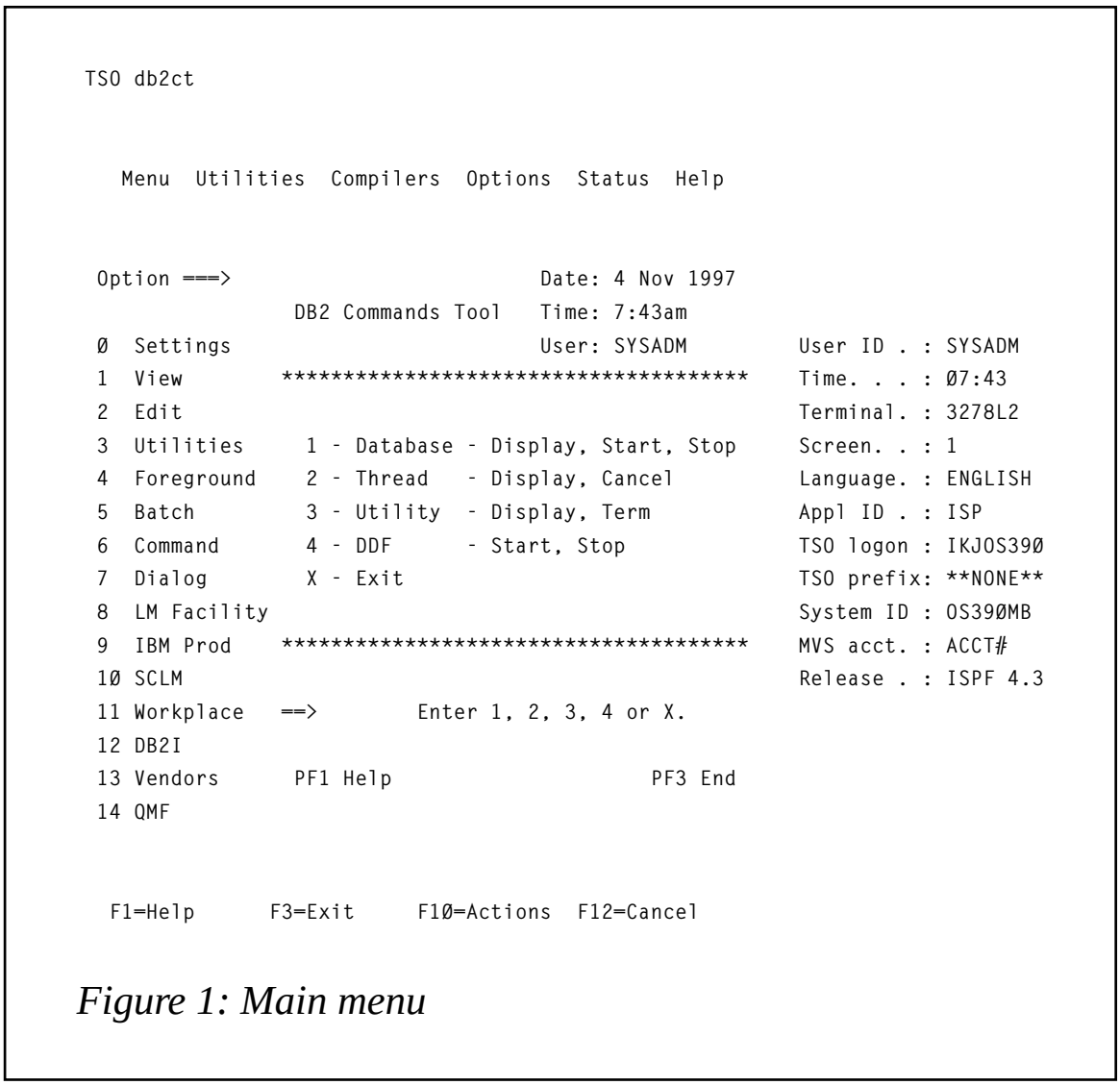

Figure 1: Main menu

DB2CT DRIVER PROCEDURE

```
/* REXX */
/* TRACE R */
 ZPFCTL = 'OFF'
 ADDRESS ISPEXEC 'VPUT (ZPFCTL) PROFILE'
 ADDRESS ISPEXEC 'ADDPOP ROW(1) COLUMN(10)'
 TOP:
 date=DATE()
 time=TIME(C)
```

```
ADDRESS ISPEXEC "DISPLAY PANEL(DB2CTPØØ)"
DO WHILE RC=Ø
   SELECT
     WHEN(X='1') THEN DO
        ADDRESS ISPEXEC REMPOP ALL
        "%DISDB"
        ADDRESS ISPEXEC 'ADDPOP ROW(1) COLUMN(1Ø)'
     END
     WHEN(X='2') THEN DO
        ADDRESS ISPEXEC REMPOP ALL
        "%DISTR"
        ADDRESS ISPEXEC 'ADDPOP ROW(1) COLUMN(1Ø)'
     END
     WHEN(X='3') THEN DO
        ADDRESS ISPEXEC REMPOP ALL
        "%DISUT"
        ADDRESS ISPEXEC 'ADDPOP ROW(1) COLUMN(1Ø)'
     END
     WHEN(X='4') THEN DO
        "%DB2DDF"
        ADDRESS ISPEXEC REMPOP ALL
        ADDRESS ISPEXEC 'ADDPOP ROW(1) COLUMN(1Ø)'
     END
     WHEN(X='X') THEN DO
        ADDRESS ISPEXEC REMPOP ALL
        EXIT
     END
     OTHERWISE RC=Ø
   END
   date=DATE()
   time=TIME(C)
   ADDRESS ISPEXEC "DISPLAY PANEL(DB2CTPØØ)"
END
EXIT
```

DISDB COMMAND PROCEDURE FOR DATABASE TABLESPACES

```
/* REXX */
/* trace r */
ZPFCTL = 'OFF'
ADDRESS ISPEXEC 'VPUT (ZPFCTL) PROFILE'
ADDRESS ISPEXEC 'TBCREATE DISDB NAMES(dbase space type part sta mes)'
CUR='db'
TOP:
RK=Ø
ADDRESS ISPEXEC 'TBDISPL DISDB PANEL(DISDBMØØ) CURSOR('CUR')'
IF RC=8 THEN RK=8
ELSE DO
       Call Check_re
```

```
        Call Check_li
        Call Check_sp
     END
DO WHILE(RK=Ø)
   Y = OUTTRAP('OUT.')
   QUEUE '-DISPLAY DATABASE('db') SPACENAM('sp') LIMIT('li') 'option''
   QUEUE 'END'
   'DSN SYSTEM('db2')'
   Y = OUTTRAP('OFF')
   Call Check
   STEP=1
   DO J = STEP TO OUT.Ø WHILE ind=Ø
      IF SUBSTR(OUT.J,1,8)='DSNT362I'
      THEN dbase=SUBSTR(OUT.J,27,8)
      IF SUBSTR(OUT.J,1,8)='--------'
      THEN CALL Detail
   END
   ADDRESS ISPEXEC 'TBTOP DISDB'
   ADDRESS ISPEXEC 'TBDISPL DISDB PANEL(DISDBMØØ) CURSOR('CUR')'
   CUR='ZCMD'
   IF RC=8 THEN RK=8
   ELSE DO
      IF cmd='S' THEN DO
         CALL Display
         "ISPEXEC ADDPOP ROW(1) COLUMN(4Ø)"
         ADDRESS ISPEXEC 'DISPLAY PANEL(DISDBM01)'
         "ISPEXEC REMPOP ALL"
      END
      IF cmd='RW' | cmd='RO' | cmd='UT' | cmd='FO'
      THEN CALL Start_db
      IF cmd='ST' THEN CALL Stop_db
      Call Check_re
      Call Check_li
      Call Check_sp
      ADDRESS ISPEXEC 'TBEND DISDB'
      ADDRESS ISPEXEC,
      'TBCREATE DISDB NAMES(dbase space type part sta mes)'
   END
END
ADDRESS ISPEXEC 'TBEND DISDB'
EXIT
Check_re:
   IF R='' THEN R='N'
   re = VERIFY(R,'YN')
   IF re > Ø THEN DO
      message='Restrict keyword only valid for Y or N.'
      ADDRESS ISPEXEC "SETMSG MSG(DISDBØØ1)"
      R='N'
      CUR='R'
      ind=1
```

```
   END
   ELSE DO
      IF R='Y'
      THEN OPTION='RESTRICT'
      ELSE OPTION=''
   END
Return
Check_li:
   ILI = VERIFY(LIM,'1234567890*')
   IF ILI > 0 THEN DO
      message='Limit keyword only valid for numeric data.'
      ADDRESS ISPEXEC "SETMSG MSG(DISDB001)"
      CUR='lim'
      ind=1
   END
   ELSE DO
      IF LIM='*' | LIM='' then LIM=50
      li=LIM+7
   END
Return
Check_sp:
   in_db = INDEX(db,'*')
   IF in_db > 0 & sp ¬= '* '
   THEN DO
      message='Spacenam keyword only valid for a single database.'
      ADDRESS ISPEXEC "SETMSG MSG(DISDB001)"
      CUR='sp'
   END
Return
Check:
   ind=0
   IF SUBSTR(OUT.1,1,8)='DSNE110E' | SUBSTR(OUT.1,1,9)='IKJ56701I',
    | SUBSTR(OUT.1,1,8)='DSNE100I'
   THEN DO
      "delstack"
      message = db2||' not valid subsystem id.'
      ADDRESS ISPEXEC "SETMSG MSG(DISDB001)"
      CUR='db2'
      ind=1
   END
   IF db='' THEN db='*'
   IF SUBSTR(OUT.1,1,8)='DSNT301I'
   THEN DO
      message = 'Invalid database 'db'.'
      ADDRESS ISPEXEC "SETMSG MSG(DISDB001)"
      CUR='db'
      ind=1
   END
   IF SUBSTR(OUT.10,1,8)='DSNT302I'
   THEN DO
```

```
      message = 'Invalid spacenan 'sp'.'
      ADDRESS ISPEXEC "SETMSG MSG(DISDBØØ1)"
      CUR='sp'
      ind=1
   END
Return
Detail:
   J=J+1
   DO I = J TO OUT.Ø WHILE(SUBSTR(OUT.I,1,7)¬='*******')
      space=SUBSTR(OUT.I,1,8)
      type=SUBSTR(OUT.I,1Ø,3)
      part=SUBSTR(OUT.I,15,4)
      mes=''
      IF dbase=dat & space=spa & part=par & cmd='S'
      THEN mes='Display'
      IF dbase=dat & space=spa & part=par & cmd='RO'
      THEN mes='Read only'
      IF dbase=dat & space=spa & part=par & cmd='RW'
      THEN mes='Read write'
      IF dbase=dat & space=spa & part=par & cmd='UT'
      THEN mes='Utility'
      IF dbase=dat & space=spa & part=par & cmd='FO'
      THEN mes='Access Force'
      IF dbase=dat & space=spa & part=par & cmd='ST'
      THEN mes='Stop'
      sta =SUBSTR(OUT.I,2Ø,18)
      pelo=SUBSTR(OUT.I,39,8)
      peli=SUBSTR(OUT.I,48,8)
      cat =SUBSTR(OUT.I,57,8)
      piece=SUBSTR(OUT.I,66,5)
      ADDRESS ISPEXEC 'TBADD DISDB'
      STEP=I
   END
   cmd="''''"
Return
Display:
   dat=dbase
   spa=space
   par=part
   in_sta = INDEX(sta,'RO')
   IF in_sta > Ø
   THEN DO
      sta1='The database,space-partition'
      sta2='is started for read only.   '
   END
   in_sta = INDEX(sta,'RW')
   IF in_sta > Ø
   THEN DO
      sta1='The database,space-partition'
      sta2='is started for read write.  '
```

```
END
in_sta = INDEX(sta,'UT')
IF in_sta > 0
THEN DO
   sta1='The database,space-partition'
   sta2='is started for utility.     '
END
in_sta = INDEX(sta,'CHKP')
IF in_sta > 0
THEN DO
   sta1='The object is in the check  '
   sta2='pending state.              '
END
in_sta = INDEX(sta,'COPY')
IF in_sta > 0
THEN DO
   sta1='The copy pending state.     '
   sta2='An image copy is required.  '
END
in_sta = INDEX(sta,'DEFER')
IF in_sta > 0
THEN DO
   sta1='The space (partition) is    '
   sta2='marked for deferred restart.'
END
in_sta = INDEX(sta,'INDBT')
IF in_sta > 0
THEN DO
   sta1='Indoubt processing is       '
   sta2='required for an object.     '
END
in_sta = INDEX(sta,'OPENF')
IF in_sta > 0
THEN DO
   sta1='The object space had an     '
   sta2='open data set failure.      '
END
in_sta = INDEX(sta,'OPENF')
IF in_sta > 0
THEN DO
   sta1='The object space had an     '
   sta2='open data set failure.      '
END
in_sta = INDEX(sta,'PSRCP')
IF in_sta > 0
THEN DO
   sta1='The index space is in a page'
   sta2='set recover pending state.  '
END
in_sta = INDEX(sta,'RECP')
```

```
IF in_sta > Ø
THEN DO
   sta1='The space object is in the  '
   sta2='recover pending state.      '
END
in_sta = INDEX(sta,'REST')
IF in_sta > Ø
THEN DO
   sta1='The table space or index    '
   sta2='space is being restarted.   '
END
in_sta = INDEX(sta,'STOP')
IF in_sta > Ø
THEN DO
   sta1='The space was implicitly    '
   sta2='stopped.                    '
END
in_sta = INDEX(sta,'STOPE')
IF in_sta > Ø
THEN DO
   sta1='The database,space-partition'
   sta2='is stopped. Log RBA problem.'
END
in_sta = INDEX(sta,'STOPP')
IF in_sta > Ø
THEN DO
   sta1='The database,space-partition'
   sta2='is in a stop pending.       '
END
in_sta = INDEX(sta,'UTRO')
IF in_sta > Ø
THEN DO
   sta1='A utility is in process,    '
   sta2='that allows only RO access. '
END
in_sta = INDEX(sta,'UTRW')
IF in_sta > Ø
THEN DO
   sta1='A utility is in process,    '
   sta2='that allows RW access.      '
END
in_sta = INDEX(sta,'UTUT')
IF in_sta > Ø
THEN DO
   sta1='A utility is in process,    '
   sta2='that allows only UT access. '
END
Y = OUTTRAP('OUT.')
option='USE'
QUEUE '-DISPLAY DATABASE('dbase') SPACENAM('space') LIMIT('li')
```

```
'option''
   QUEUE 'END'
   'DSN SYSTEM('db2')'
   Y = OUTTRAP('OFF')
   DO I=1 TO OUT.Ø
      IF dbase=dat & spa=SUBSTR(OUT.I,1,8) & par=SUBSTR(OUT.I,15,4) then
do
         con=SUBSTR(OUT.I,39,8)
         cor=SUBSTR(OUT.I,48,12)
         use=SUBSTR(OUT.I,61,8)
      END
   END
   Y = OUTTRAP('OUT.')
   option='LOCKS'
   QUEUE '-DISPLAY DATABASE('dbase') SPACENAM('space') LIMIT('li')
'option''
   QUEUE 'END'
   'DSN SYSTEM('db2')'
   Y = OUTTRAP('OFF')
   DO I = 1 TO OUT.Ø
      IF dbase=dat & spa=SUBSTR(OUT.I,1,8) & par=SUBSTR(OUT.I,15,4) then
         lock=SUBSTR(OUT.I,61,9)
   END
   Y = OUTTRAP('OUT.')
   option='CLAIMERS'
   QUEUE '-DISPLAY DATABASE('dbase') SPACENAM('space') LIMIT('li')
'option''
   QUEUE 'END'
   'DSN SYSTEM('db2')'
   DO I = 1 TO OUT.Ø
      IF dbase=dat & spa=SUBSTR(OUT.I,1,8) & par=SUBSTR(OUT.I,15,4) then
         cla=SUBSTR(OUT.I,61,8)
   END
   Y = OUTTRAP('OFF')
Return
Start_db:
   dat=dbase
   spa=space
   par=part
   IF cmd = 'FO' THEN cmd = 'FORCE'
   IF part ¬= ' '
   THEN option='PART('||part||') ACCESS('cmd')'
   ELSE option='ACCESS('cmd')'
   Y = OUTTRAP('OUT.')
   QUEUE '-START DATABASE('dbase') SPACENAM('space') 'option''
   QUEUE 'END'
   'DSN SYSTEM('db2')'
   Y = OUTTRAP('OFF')
   IF cmd = 'FORCE' THEN cmd = 'FO'
Return
```

```
Stop_db:
   dat=dbase
   spa=space
   par=part
   IF part ¬= ' '
   THEN option='PART('||part||') AT(COMMIT)'
   ELSE option=' AT(COMMIT)'
   Y = OUTTRAP('OUT.')
   QUEUE '-STOP DATABASE('dbase') SPACENAM('space') 'option''
   QUEUE 'END'
   'DSN SYSTEM('db2')'
   Y = OUTTRAP('OFF')
Return
```

DISTR DDF THREAD PROCEDURE

```
/* REXX */
/* trace r */
ZPFCTL = 'OFF'
ADDRESS ISPEXEC 'VPUT (ZPFCTL) PROFILE'
ADDRESS ISPEXEC 'VGET (db2) PROFILE'
RK=Ø
DO WHILE(RK=Ø)
   IF db2='' THEN Call Check
   Y = OUTTRAP('OUT.')
   QUEUE '-DISPLAY THREAD(*) LOCATION(*)'
   QUEUE 'END'
   'DSN SYSTEM('db2')'
   Y = OUTTRAP('OFF')
   Call Check
   ADDRESS ISPEXEC 'TBCREATE DISTR NAMES(row, token, luwid, mess)'
   DO I = 4 TO OUT.Ø - 2 BY 1
      IF SUBSTR(OUT.I,1,1)<>' ' THEN DO
         J=I+1
         token=STRIP(WORD(TRANSLATE(OUT.J,' ','='),2),,' ')
         row = OUT.I||'   '||token
         luwid = OUT.J
         IF data = luwid
         THEN mess = 'Display'
         ELSE mess = ''
         /* DATA = SUBSTR(OUT.J,WORDINDEX(OUT.J,2)) */
         ADDRESS ISPEXEC 'TBADD DISTR'
      END
   END
   ADDRESS ISPEXEC 'TBTOP DISTR'
   ADDRESS ISPEXEC 'TBDISPL DISTR PANEL(DISTRMØØ)'
   IF cmd='S'
   THEN data = luwid
   ELSE data = ''
```

```
   IF cmd='C'
   THEN DO
        Y = OUTTRAP('OUT.')
        QUEUE '-CANCEL DDF THREAD('token')'
        QUEUE 'END'
        'DSN SYSTEM('db2')'
        Y = OUTTRAP('OFF')
        Call Check
        if ind=Ø
        THEN DO
             message = SUBSTR(OUT.1,12)
             ADDRESS ISPEXEC "SETMSG MSG(DISDBØØ1)"
        END
   END
   IF cmd='A'
   THEN DO
        QUEUE '-DISPLAY THREAD(*) LUWID('token') DETAIL'
        QUEUE 'END'
        'DSN SYSTEM('db2')'
        Call Check
   END
   IF RC=8 THEN RK=8
   ADDRESS ISPEXEC 'TBEND DISTR'
END
Check:
   ind=Ø
   IF SUBSTR(OUT.1,1,8)='DSNE11ØE' | SUBSTR(OUT.1,1,9)='IKJ567Ø1I',
    | SUBSTR(OUT.1,1,8)='DSNE1ØØI'
   THEN DO
      "delstack"
      message = db2||' not valid subsystem id.'
      ADDRESS ISPEXEC "SETMSG MSG(DISDBØØ1)"
      ind=1
   END
Return
EXIT
```

DISUT COMMAND PROCEDURE FOR UTILITY

```
/* REXX */
/* trace r */
ZPFCTL = 'OFF'
ADDRESS ISPEXEC 'VPUT (ZPFCTL) PROFILE'
ADDRESS ISPEXEC 'VGET (db2) PROFILE'
RK=Ø
DO WHILE(RK=Ø)
   IF db2='' THEN Call Check
   Y = OUTTRAP('OUT.')
   QUEUE '-DISPLAY UTILITY(*)'
```

```
QUEUE 'END'
'DSN SYSTEM('db2')'
Y = OUTTRAP('OFF')
Call Check
ADDRESS ISPEXEC 'TBCREATE DISUT NAMES(Uid, Id, Util, Pha, Cou, Sta)'
DO I = 1 TO OUT.Ø - 1 BY 6 WHILE (OUT.Ø > 5)
   Id  = WORD(OUT.I,7)
   I1=I+1
   Uid = WORD(OUT.I1,3)
   I2=I+3
   Util =WORD(OUT.I2,3)
   I3=I+4
   Pha = WORD(OUT.I3,3)
   Cou = WORD(OUT.I3,6)
   I4=I+5
   Sta = WORD(OUT.I4,3)
   ADDRESS ISPEXEC 'TBADD DISUT'
END
ADDRESS ISPEXEC 'TBTOP DISUT'
ADDRESS ISPEXEC 'TBDISPL DISUT PANEL(DISUTMØØ)'
IF ans='YES'
THEN DO
     Y = OUTTRAP('OUT.')
     QUEUE '-TERM UTILITY (*)'
     QUEUE 'END'
     'DSN SYSTEM('db2')'
     Y = OUTTRAP('OFF')
     cmd=''
     Call Check
     if ind=Ø then do
        message = '-TERM UTILITY (*)'
        ADDRESS ISPEXEC "SETMSG MSG(DISDBØØ1)"
     end
END
IF cmd='C'
THEN DO
     Y = OUTTRAP('OUT.')
     QUEUE '-TERM UTILITY ('uid')'
     QUEUE 'END'
     'DSN SYSTEM('db2')'
     Y = OUTTRAP('OFF')
     Call Check
     if ind=Ø then do
        text=WORD(OUT.4,6)||' '||WORD(OUT.4,7)
        IF text='NORMAL COMPLETION'
        THEN message = '-TERM UTILITY ('||uid||') '||text
        ELSE message = OUT.1||' '||OUT.2
        ADDRESS ISPEXEC "SETMSG MSG(DISDBØØ1)"
     end
END
```

```
   IF RC=8 THEN RK=8
   ADDRESS ISPEXEC 'TBEND DISUT'
END
Check:
   ind=Ø
   IF SUBSTR(OUT.1,1,8)='DSNE11ØE' | SUBSTR(OUT.1,1,9)='IKJ567Ø1I',
    | SUBSTR(OUT.1,1,8)='DSNE1ØØI'
   THEN DO
      "delstack"
      message = db2||' not valid subsystem id.'
      ADDRESS ISPEXEC "SETMSG MSG(DISDBØØ1)"
      ind=1
   END
Return
EXIT
```

DB2DDF START, STOP DDF PROCEDURE

```
/* REXX */
/* TRACE R */
 ZPFCTL = 'OFF'
 ADDRESS ISPEXEC 'VPUT (ZPFCTL) PROFILE'
 ADDRESS ISPEXEC 'ADDPOP ROW(1) COLUMN(4Ø)'
 DO WHILE RC=Ø
    Y = OUTTRAP('OUT.')
    SELECT
      WHEN(X='1') THEN DO
         QUEUE '-START DDF'
         QUEUE 'END'
         'DSN SYSTEM('db2')'
      END
      WHEN(X='2') THEN DO
         QUEUE '-STOP DDF MODE(QUIESCE)'
         QUEUE 'END'
         'DSN SYSTEM('db2')'
      END
      WHEN(X='3') THEN DO
         QUEUE '-STOP DDF MODE(FORCE)'
         QUEUE 'END'
         'DSN SYSTEM('db2')'
      END
      OTHERWISE RC=Ø
    END
    Y = OUTTRAP('OFF')
    Call Check
    IF ind=Ø THEN DO
       message = SUBSTR(OUT.1,12)
       ADDRESS ISPEXEC "SETMSG MSG(DISDBØØ1)"
    END
    ADDRESS ISPEXEC "DISPLAY PANEL(DB2DDFØØ)"
```

```
 END
 Check:
    ind=Ø
    IF SUBSTR(OUT.1,1,8)='DSNE11ØE' | SUBSTR(OUT.1,1,9)='IKJ567Ø1I',
     | SUBSTR(OUT.1,1,8)='DSNE1ØØI'
    THEN DO
       "delstack"
       message = db2||' not valid subsystem id.'
       ADDRESS ISPEXEC "SETMSG MSG(DISDBØØ1)"
       ind=1
    END
 Return
EXIT
```

DB2CTP00 MAIN MENU

```
)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(41,16) EXPAND ($$)
+!                    + Date:|date       +
+! DB2 Commands Tool + Time:|time      +
+!                    + User: &zuser
/ ************************************
+
+  [row1                               +
+  [row2                               +
+  [row3                               +
+  [row4                               +
+  [row5                               +
+
/ ************************************
+
+@==>+ _X+  #msg                       +
+
+! PF1 Help +                ! PF3 End +
)INIT
  &row1= '1 - Database - Display, Start, Stop'
  &row2= '2 - Thread   - Display, Cancel'
  &row3= '3 - Utility  - Display, Term'
  &row4= '4 - DDF      - Start, Stop'
  &row5= 'X - Exit'
  IF (&X = 1,2,3,4,X)
```

```
      &msg = ''
  ELSE
      .ATTR (msg) = 'COLOR (RED)'
      &msg = 'Enter 1, 2, 3, 4 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)'
  IF (&X = 4)
      .ATTR (row4) = 'COLOR (YELLOW) CAPS(ON)'
)PROC
  IF (.PFKEY = PFØ3) &PF3 = EXIT
)END
```

DISDBM00 DISPLAY DATABASE_TABLESPACE PANEL

```
)Attr Default(%+_)
   | type(text)   intens(high) caps(on ) color(yellow)
   $ type(output) intens(high) caps(off) color(yellow)
   ? type(text)   intens(high) caps(on ) color(green) hilite(reverse)
   # type(text)   intens(high) caps(off) hilite(reverse)
   } type(text)   intens(high) caps(off) color(white)
   [ type( input) intens(high) caps(on ) just(left )
   ! type( input) intens(high) caps(on ) just(left ) pad('''')
   ¬ type(output) intens(low ) caps(off) just(asis ) color(turquoise)
)Body  Expand(//)
%-/-/- ? Display database status +%-/-/-
%Command ===>_zcmd                                        / /%Scroll
===>_amt +
+SSID[db2 + Database:[db      + Spacenam:[sp      + Limit:[lim +
Restrict:[R+
+-------------------------------------------------------------------------
--------
+Valid cmd:|S+Display|RW+Read Write|RO+Read
Only|UT+Utility|FO+Force|ST+Stop
+-------------------------------------------------------------------------
--------
#cmd+ #Database+ #Spacenam+ #Type+ #Part+ #Status+
)Model
!z  + ¬z       + ¬z       + ¬z   + ¬z   + ¬z                 + $z
+
)Init
  .ZVARS = '(cmd dbase space type part sta mes)'
  &amt = PAGE
)Reinit
)Proc
  VPUT (db2 db sp lim R) PROFILE
)End
```

DSDBM01 DISPLAY DATABASE-TABLESPACE PANEL (ADDITIONAL INFORMATION)

```
)Attr Default(%+_)
   | type(text) intens(high) caps(on )  color(white) hilite(reverse)
   / type(text) intens(high) caps(off ) color(yellow)
)body window(31,2Ø)
|  Detail display Information
+
+Database  %&dbase
+Name      %&space  +
+Type      %&type+
+Part     %&part+
+Status    %&sta              +
+/&sta1                      +
+/&sta2                      +
+Phyerrlo  %&pelo   +
+Phyerrhi  %&peli   +
+Catalog   %&cat    +
+Piece     %&piece+
+Connid    %&con    +
+Corrid    %&cor        +
+Userid    %&use    +
+Lockinfo  %&lock    +
+Claiminfo %&cla    +
+
          | FP3 End +
)init
)proc
)end
```

DISTRM00 DISPLAY PANEL THREAD

```
)Attr Default(%+_)
   | type(text)   intens(high) caps(on ) color(yellow)
   $ type(output) intens(high) caps(off) color(yellow)
   ? type(text)   intens(high) caps(on ) color(green) hilite(reverse)
   # type(text)   intens(high) caps(off) hilite(reverse)
   } type(text)   intens(high) caps(off) color(white)
   [ type( input) intens(high) caps(on ) just(left )
   ! type( input) intens(high) caps(on ) just(left ) pad('-')
   ¬ type(output) intens(low ) caps(off) just(asis ) color(turquoise)
)Body  Expand(//)
%-/-/- ? Display DDF thread activity +%-/-/-
%Command ===>_zcmd                                      / /%Scroll
===>_amt +
+SSID[db2 +
+-------------------------------------------------------------------
```

```
--------
+Valid cmd:|S+Display|C+Cancel
+-------------------------------------------------------------------------
--------
   ¬data
+
#cmd#Name    #St#A # Req#Id          #Authid  #Plan    #Asid  #Luw  +
)Model
 !z+¬z                                                              +$z
+
)Init
  .ZVARS = '(cmd row mess)'
  &amt = PAGE
  &cmd = ''
)Reinit
)Proc
  VPUT (db2) PROFILE
)End
```

DISUTM00 DISPLAY UTILITY PANEL

```
)Attr Default(%+_)
   | type(text)   intens(high) caps(on ) color(yellow)
   $ type(output) intens(high) caps(off) color(yellow)
   ? type(text)   intens(high) caps(on ) color(green) hilite(reverse)
   # type(text)   intens(high) caps(off) hilite(reverse)
   } type(text)   intens(high) caps(off) color(white)
   [ type( input) intens(high) caps(on ) just(left )
   ! type( input) intens(high) caps(on ) just(left ) pad('-')
   ¬ type(output) intens(low ) caps(off) just(asis ) color(turquoise)
)Body  Expand(//)
%-/-/- ? Utility information +%-/-/-
%Command ===>_zcmd                                      / /%Scroll
===>_amt +
+SSID[db2 +                                           |Terminate all
utility [ans+
+-------------------------------------------------------------------------
--------
+Valid cmd:|C+Cancel
+-------------------------------------------------------------------------
--------
#cmd#Utilid            #Userid  #Utility   #Phase        #Count
#Status  +
)Model
 !z+¬z               +¬z      +¬z        +¬z           +¬z        +¬z
+
)Init
  .ZVARS = '(cmd uid id util pha cou sta)'
  &ans = NO
```

```
  &amt = PAGE
  &cmd = ''
)Reinit
)Proc
  VPUT (db2) PROFILE
)End
```

DB2DDF00 START, STOP PANEL

```
)Attr Default(%+_)
   | type(text) intens(high) caps(on )  color(white) hilite(reverse)
   # type(text) intens(high) caps(off ) color(green)
   | type(text) intens(high) caps(off ) color(red)
   / type(text) intens(high) caps(off ) color(yellow)
   \ type(output) intens(high) caps(off ) color(yellow)
   [ type(input) intens(high) caps(on ) just(left ) pad('_')
   @ type(text) intens(high) color(red) caps(off) hilite(reverse)
)body window(27,13)
|        DDF Action
+
+SSID[db2 +
+
+ ------------------------
# 1-START DDF
+
| 2-STOP DDF MODE(QUIESCE)
| 3-STOP DDF MODE(FORCE)
+ ------------------------
+ ==>[X+\msg              +
+
        | FP3 End +
)init
   IF (&X = 1,2,3)
       &msg = ''
   ELSE
       &msg = 'Enter 1, 2, or 3.'
)proc
    VPUT (db2) PROFILE
)end
```

DISDB00 DB2CT MESSAGE

```
DISDBØØ1          .ALARM = YES  .WINDOW=NORESP .ALARM = YES
'&message'
```

Bernard Zver
Database Administrator
Informatika Maribor (Slovenia)

DB2 terminate utility

INTRODUCTION

The DB2 command TERM UTILITY terminates the execution of a utility job step and releases all resources associated with the step. When executing, a utility does not terminate until it checks to see that TERM was issued. Active utilities perform this check periodically. If the utility is stopped, all its resources are released by the TERM command. This command can be issued from an MVS console, a DSN session, DB2I panels, DB2 commands and DB2 utilities, a CICS terminal, or a program using the instrumentation facility interface.

To execute this command, the primary or some secondary authorization ID of the process must be the ID that originally submitted the utility job, or the privilege set of the process must include one of the following: DBMAINT, DBCTRL, DBADM, SYSOPR, SYSCTRL, or SYSADM authority.

This command is used by DBAs, application programmers, operations staff, or any one who runs a DB2 utility and subsequently wishes to terminate it. To terminate a utility, we must know the utility-id. For this, we have to first issue a DIS UTIL(*) command to find the utility-id of the utility to terminate, and then issue the TERM command. If we have to terminate more than one utility, we must note down the utility-ids of all the utilities and then terminate them one by one – a tedious procedure.

To make this job easier, I have written an ISPF function that lists all utilities executing/stopped in a given subsystem and provides selective termination from the list.

The REXX EXEC TERMUTIL performs a DIS UTIL(*) command for the subsystem specified and captures the output in an array. It then parses through this data and extracts the tedious, utility name, utility phase, utility status, and auth-id for each utility. These are displayed on a panel, and the panel PTRMUTL allows the user to browse

through the list. Entering a ‘T’ in the line command field will cause the utility displayed on that line to be terminated.

TERMUTIL EXEC

```
/* REXX ***************************************************************
  ADDRESS "ISPEXEC"
  "LIBDEF ISPPLIB DATASET ID ( < dataset for ispplib > )"
  "LIBDEF ISPTLIB DATASET ID ( < dataset for isptlib > )"
  ADDRESS "ISPEXEC" "TBCREATE UTLLIST"||,
      " KEYS(UTILID UTILNAME UTILPHS UTILSTS UTLUSR)"||,
      " NOWRITE REPLACE"
  ADDRESS "ISPEXEC" "TBDISPL UTLLIST PANEL(PTRMUTL)"
  ADDRESS "ISPEXEC" "VPUT (ZZSSID) PROFILE"
  CALL BLD_UTLLIST
  ADDRESS "ISPEXEC" "TBDISPL UTLLIST PANEL(PTRMUTL)"
  IF ZTDSELS > Ø THEN
    DO
      SELCOUNT = ZTDSELS
      DO J = 1 TO SELCOUNT
        CALL GET_ROW
        CALL PROCESS_UTL
        IF J < SELCOUNT THEN ADDRESS "ISPEXEC" "TBDISPL UTLLIST"
      END
  END
  ADDRESS "ISPEXEC"
  "LIBDEF ISPPLIB "
  "LIBDEF ISPTLIB "
  EXIT
/*********************************************************************/
/* BLD_UTLLIST                                                       */
/*********************************************************************/
BLD_UTLLIST:
    DUMMY = OUTTRAP("OUTPUT_LINE.","*")
    QUEUE "-DIS UTIL(*)"
    QUEUE "END"
    ADDRESS TSO
    "DSN SYSTEM("ZZSSID")"
    IF RC > Ø THEN
      DO
        SAY 'RC='||RC
        DO I = 1 TO OUTPUT_LINE.Ø
          SAY OUTPUT_LINE.I
        END
        RETURN
      END
    N = Ø
```

```
    DO I = 1 TO OUTPUT_LINE.Ø
        CALL PARSE_THREADS
    END
    DO I = 1 TO N
      ACTION = ''
      UTILID = TUTILID.I
      UTILNAME = TUTILNAME.I
      UTILPHS = TUTILPHS.I
      UTILSTS = TUTILSTS.I
      UTLUSR  = TUTLUSR.I
      ADDRESS "ISPEXEC" "TBADD UTLLIST"
      IF RC > Ø  THEN
        DO
          SAY 'TBADD ERROR '||RC
          ADDRESS "ISPEXEC" "TBEND UTLLIST"
          EXIT
        END
    END
  ADDRESS "ISPEXEC" "TBSORT UTLLIST "||,
                    "FIELDS(UTILID,C,A,UTILNAME,C,A,UTILPHS,C,A)"
  ADDRESS "ISPEXEC" "TBTOP UTLLIST"
  IF RC =4 THEN
    DO
      SAY 'TBTOP ERROR '||RC
      ADDRESS "ISPEXEC" "TBEND UTLLIST"
      EXIT
    END
RETURN;
/*****************************************************************/
/* PARSE_THREADS                                                 */
/*****************************************************************/
PARSE_THREADS:
POS=Ø
POS=POS('DSNUGDIS - USERID = ',OUTPUT_LINE.I)
IF POS > Ø THEN DO
  N=N+1
  POS=POS+2Ø
      TUTLUSR.N  = SUBWORD(SUBSTR(OUTPUT_LINE.I,POS),1,1)
      RETURN
  END
POS=POS('UTILID = ',OUTPUT_LINE.I)
IF POS > Ø THEN DO
  POS=POS+9
      TUTILID.N  = SUBWORD(SUBSTR(OUTPUT_LINE.I,POS),1,1)
      RETURN
  END
POS=POS('UTILITY = ',OUTPUT_LINE.I)
IF POS > Ø THEN DO
  POS=POS+1Ø
```

```
      TUTILNAME.N  = SUBWORD(SUBSTR(OUTPUT_LINE.I,POS),1,1)
      RETURN
  END
POS=POS('PHASE = ',OUTPUT_LINE.I)
IF POS > Ø THEN DO
  POS=POS+8
      TUTILPHS.N  = SUBWORD(SUBSTR(OUTPUT_LINE.I,POS),1,1)
      RETURN
  END
POS=POS('STATUS = ',OUTPUT_LINE.I)
IF POS > Ø THEN DO
  POS=POS+9
      TUTILSTS.N  = SUBWORD(SUBSTR(OUTPUT_LINE.I,POS),1,1)
      RETURN
  END

RETURN;
/*****************************************************************/
/* GET_ROW                                                       */
/*****************************************************************/
GET_ROW:
  ADDRESS "ISPEXEC" "TBGET UTLLIST"
RETURN;
/*****************************************************************/
/* PROCESS_UTL                                                   */
/*****************************************************************/
PROCESS_UTL:
    PARSE UPPER VAR ACTION ACT
    IF ACT = 'T' THEN
      DO
        QUEUE "-TERM   UTIL("||UTILID||")"
        QUEUE "END"
        ADDRESS TSO "DSN SYSTEM("ZZSSID")"
      END
RETURN;
/**************   PANEL PTRMUTL     ********************/
)ATTR
/*************************************************************************/
/*                                                                       */
/* PTRMUTL - TERMINATE DB2 UTILITIES                                     */
/*                                                                       */
/*************************************************************************/
   + TYPE(TEXT)   INTENS(LOW)   COLOR(BLUE)         SKIP(ON)
   % TYPE(TEXT)   INTENS(HIGH)  COLOR(WHITE)        SKIP(ON)
   $ TYPE(TEXT)   INTENS(HIGH)  COLOR(RED)          SKIP(ON)
   # TYPE(OUTPUT) INTENS(HIGH)  COLOR(TURQUOISE)    CAPS(ON)
)BODY CMD(C)
%--------------------+DB2 UTILITIES TERMINATION %--------------------
%OPTION ===_C                                              %SCR-_AMT +
```

```
+                                                    +DB2 SSID: _Z   +
+
% CMD    UTILITY ID        TYPE       PHASE          STATUS    USERID
%-------------------------------------------------------------------+
)MODEL
_ACTION+#UTILID          +#UTILNAME +#UTILPHS      +#UTILSTS +#UTLUSR +
)INIT
 .ZVARS = '(ZZSSID)'
)PROC
  VER (&C LIST,END,' ',CAN,CANCEL)
  VER (&ZZSSID NONBLANK LIST,DDB3,DST1,DST2,TRN1)
)END
```

Sharad K Pande
DB2 DBA (USA)

REXX extensions for DB2

REXXPLUS/MVS is a set of functions and subroutines that extend IBM REXX. These functions are built upon the standard facilities of REXX, and they have the same syntax as REXX.

REXXPLUS/MVS is interfaced with IBM DB2. Requests to DB2 are made under TSO using the standard SQL language through the ADDRESS DB2 statement.

INSTALLATION

To install REXXPLUS/MVS, follow the instructions below:

1 Assemble (or compile, for the COBOL program) all the programs in a library (eg MY.LIB) with the RENT attribute, except for IRXFLOC and $IRXCNV2, which must be non-reentrant, non-reusable; and except for $IRXTERM, which must be link-edited reentrant, AC=1, in your linklist.

  The $IRXTERM program is a clean-up routine. The $IRXDB2H program is a DB2 program, and must be assembled with the appropriate JCL. You must create a plan under all the DB2s for which you want to use REXXPLUS/MVS. The input data for the BIND could be:

```
BIND PLAN($IRXDB2H) MEM($IRXDB2H) ACT(REP) ISOLATION(CS) LIB(...)
```

2 Copy the SYS1.SAMPLIB(TSOREXX1) member into one of your own libraries, and update it by replacing the following line:

```
+-------------------------------------------------------------------+
! MODNAMET_EXECTERM DC CL8'       '                                  !
+-------------------------------------------------------------------+
```

with:

```
+-------------------------------------------------------------------+
! MODNAMET_EXECTERM DC CL8'$IRXTERM'                                 !
+-------------------------------------------------------------------+
```

Then assemble this new member, and put the load module into your library, MY.LIB, with the name IRXPARMS (this name is fixed by IBM), with the AC=0 and RENT attributes.

3 Include the library MY.LIB in the STEPLIB of your TSO log-on procedure and the STEPLIB of all batch jobs calling REXXPLUS/MVS.

Notice that the clean up routine, $IRXTERM, is called only by the address spaces that have the modified IRXPARMS in their STEPLIB. $IRXTERM is not called from standard IBM REXX routines that have the standard IRXPARMS, so installing $IRXTERM in your linklist does not have any effect on your standard REXX.

DB2 PROCESSING

$DB2INST

```
+-------------------------------------------------------------------+
! RC=$DB2INST(type_col_name, conversion, var_descr, var_col, env_name) !
+-------------------------------------------------------------------+
```

This function sets up the DB2 environment and gives various processing options to REXXPLUS/MVS.

Argument 1 ‘type_col_name’ allows you to specify the name of the variables that will receive the values of the columns during a FETCH. Possible values for this argument are:

S (means Standard). With this option the names of the variables will

be the same as the names of the DB2 columns, except for nameless columns for which the names of the variables are given by the fourth argument. ‘S’ is the default value.

F (means Force). With this option, the names of the variables are given by the fourth argument, even for a column having a name.

Argument 2 ‘conversion’ allows you to convert (or not) the data coming from DB2. Possible values for this argument are:

C (means Conversion). This option specifies that data of type INTEGER, SMALLINT, DECIMAL, and FLOAT must be converted to extended decimal. ‘C’ is the default value.

N (means No conversion). This option specifies that the data will not be converted.

Argument 3 ‘var_descr’ specifies the prefix to the names of variables that will receive the columns’ descriptions during a DECLARE CURSOR.

The default value is ‘$COLNT.’, so that the $COLNT.1, $COLNT.2 (and so on) variables will contain the description for the first column, the second column, etc. The $COLNT.0 variable will contain the number of columns.

See description of *DECLARE CURSOR* for more details on column descriptions.

Argument 4 ‘var_col’ specifies the prefix of the names of the variables that will receive the content of the columns for each line during a FETCH, when the column has no name, or when the argument ‘type_col_name’ is ‘F’. The default value is ‘$COL’, so that the $COL1, $COL2 (and so on) variables will contain the values of the columns.

When argument 1 is ‘S’, the number added behind the prefix is the number of the column. For example, if the first column has the name ‘EMPL_NAME’ and the second column has no name, then the name of the variable for column 1 will be ‘EMPL_NAME’, and the name of the variable for column 2 will be $COL2, although the $COL1 variable does not exist.

Argument 5 ‘env_name’ allows you to specify the name of the REXX environment under which DB2 is called. The default value is ‘DB2’, so a DB2 call should be made by way of the ‘ADDRESS DB2’ instruction.

A call to the $DB2INST function is mandatory before ADDRESS DB2 can be used. When the default values are acceptable, you can call this function as:

```
CODE=$DB2INST().
```

Or for example:

```
CODE=$DB2INST('F',,'DESCR','COL')
```

SQL clauses

```
+-------------------------------------------------------------------------+
! ADDRESS DB2 "clause"                                                     !
+-------------------------------------------------------------------------+
```

DB2 calls are made using the REXX ADDRESS DB2 instruction. The value ‘DB2’ can be modified by the $DB2INST function.

Recognized SQL clauses are those described below. Their syntax is described in IBM documentation, except that REXXPLUS/MVS does not accept host variables or descriptors, which are unacceptable in REXX.

After the execution of any SQL clause, the SQLCODE and SQLSTATE variables contain return codes from DB2. See IBM documentation for the meaning of these codes.

ALTER clause

```
+-------------------------------------------------------------------------+
! "ALTER ..."                                                             !
+-------------------------------------------------------------------------+
```

See IBM documentation for the syntax of this clause.

CLOSE clause

```
+-------------------------------------------------------------------------+
! "CLOSE xx"                                                              !
+-------------------------------------------------------------------------+
```

See IBM documentation for the syntax of this clause.

'xx' must be the name of a cursor used in the DECLARE CURSOR clause.

COMMIT clause

```
+-------------------------------------------------------------------------+
! "COMMIT ..."                                                            !
+-------------------------------------------------------------------------+
```

See IBM documentation for the syntax of this clause.

CONN clause

```
+-------------------------------------------------------------------------+
! "CONN DB2_name"                                                         !
+-------------------------------------------------------------------------+
```

This clause is not part of the SQL language. It allows you to connect to a DB2.

On return, the SQLCODE and SQLSTATE variables are not assigned a value.

Assigned variables are RC, which contains the return code, and REASON, which contains the reason code.

CREATE clause

```
+-------------------------------------------------------------------------+
! "CREATE ..."                                                            !
+-------------------------------------------------------------------------+
```

See IBM documentation for the syntax of this clause.

DECLARE CURSOR clause

```
+-------------------------------------------------------------------------+
! "DECLARE xx CURSOR <WITH HOLD> FOR ..."                                 !
+-------------------------------------------------------------------------+
```

See IBM documentation for the syntax of this clause and its usage.

Four cursors are at your disposal:

- Two cursors, C1 and C2, without the 'WITH HOLD' option.

- Two cursors, H1 and H2, with the ‘WITH HOLD’ option.

You can’t use a Cx cursor with the ‘WITH HOLD’ option, nor a Hx cursor without the ‘WITH HOLD’ option. You can’t use any other cursor name.

The words ‘DECLARE xx CURSOR <WITH HOLD> FOR ’ must be separated by only one blank (the following SELECT clause has no restriction except for the length).

After execution of this clause, the $COLNT.i variables contain the description of the selected column (the names of these variables can be modified by a $DB2INST function).

Each $COLNT.i variable contains the following data:

- Name of the columns. When the column has no name, this field contains an asterisk (*).
- Type of the column (CHAR, INTEGER, etc) followed by its length enclosed by parenthesis.
- The ‘NOT NULL’ string if the column has the NOT NULL attribute.

Here is an example of a column description:

```
EMP_NAME CHAR(00032) NOT NULL
```

DELETE clause

```
+-------------------------------------------------------------------------+
! "DELETE ..."                                                            !
+-------------------------------------------------------------------------+
```

See IBM documentation for the syntax of this clause.

DISC clause

```
+-------------------------------------------------------------------------+
! "DISC <SYNC>"                                                           !
! or                                                                      !
! "DISC ABRT"                                                             !
+-------------------------------------------------------------------------+
```

This clause is not part of the SQL language. It allows you to disconnect from a DB2.

On return, the SQLCODE and SQLSTATE variables are not assigned a value.

Assigned variables are RC, which contains the return code, and REASON, which contains the reason code.

The clause DISC or DISC SYNC is the normal way of disconnecting. The clause DISC ABRT allows you to disconnect and to disregard the modifications made since the last commit point.

DROP clause

```
+-----------------------------------------------------------------------+
! "DROP ..."                                                            !
+-----------------------------------------------------------------------+
```

See IBM documentation for the syntax of this clause.

FETCH clause

```
+-----------------------------------------------------------------------+
! "FETCH xx"                                                            !
+-----------------------------------------------------------------------+
```

‘xx’ must be the name of a cursor used in the clause DECLARE CURSOR.

See IBM documentation for the syntax of this clause.

After execution, the variables, of which the names are the column names, contain the values of the columns for the current line. So, if the name of column 1 is EMP_NAME and if the current line contains ‘SMITH’ for this column, then the EMP_NAME variable will contain the value ‘SMITH’. See the $DB2INST function for more details about the column names.

The NULL or NOT NULL attribute is set in the same variable name, prefixed by the ‘NULL_’ string. The value is 0 if the field is NOT NULL, or 1 if the field is NULL. So, in the above example, the NULL_EMP_NAME variable contains the value 0. Should a value be null, the variable containing the column value would have a null length. Therefore, if the column EMP_NUM is NULL for this line, then the NULL_EMP_NUM variable will contain 1, and the EMP_NUM variable will contain the null string ‘’.

GRANT clause

```
+-------------------------------------------------------------------+
! "GRANT ..."                                                       !
+-------------------------------------------------------------------+
```

See IBM documentation for the syntax of this clause.

INSERT clause

```
+-------------------------------------------------------------------+
! "INSERT ..."                                                      !
+-------------------------------------------------------------------+
```

See IBM documentation for the syntax of this clause.

LOCK clause

```
+-------------------------------------------------------------------+
! "LOCK ..."                                                        !
+-------------------------------------------------------------------+
```

See IBM documentation for the syntax of this clause.

OPEN clause

```
+-------------------------------------------------------------------+
! "OPEN xx"                                                         !
+-------------------------------------------------------------------+
```

‘xx’ must be the name of a cursor used in the clause DECLARE CURSOR.

See IBM documentation for the syntax of this clause.

ROLLBACK clause

```
+-------------------------------------------------------------------+
! "ROLLBACK ..."                                                    !
+-------------------------------------------------------------------+
```

See IBM documentation for the syntax of this clause.

REVOKE clause

```
+-------------------------------------------------------------------+
! "REVOKE ..."                                                      !
+-------------------------------------------------------------------+
```

See IBM documentation for the syntax of this clause.

SET CURRENT SQLID clause

```
+-------------------------------------------------------------------------+
! "SET CURRENT SQLID ..."                                                 !
+-------------------------------------------------------------------------+
```

See IBM documentation for the syntax of this clause.

UPDATE clause

```
+-------------------------------------------------------------------------+
! "UPDATE ..."                                                            !
+-------------------------------------------------------------------------+
```

See IBM documentation for the syntax of this clause.

SAMPLE DB2 PROGRAM

Here is a program that calls DB2. It asks the user for a SELECT clause, then displays which columns it will receive, and then displays some lines.

```
/* REXX */
RC=$DB2INST('F','C',,'COL.')
ADDRESS DB2
"CONN DB2T"
 IF RC¬=Ø THEN DO
                SAY 'CONNECTION TO DB2T IMPOSSIBLE'; RETURN
               END

  SAY 'GIVE YOUR SELECT'
  PULL SELECT
  DO I=1 TO 999 WHILE SELECT¬=''
  "DECLARE C1 CURSOR FOR" SELECT
  IF SQLSTATE¬='ØØØØØ' THEN DO; CALL ERR; LEAVE I; END
  SAY 'YOU WILL OBTAIN THE' $COLNT.Ø 'FOLLOWING COLUMNS:'
  DO I=1 TO $COLNT.Ø
     SAY $COLNT.I
  END
  "OPEN C1"
  IF SQLSTATE¬='ØØØØØ' THEN DO; CALL ERR; LEAVE I; END
  SAY 'HOW MANY LINES DO YOU WANT ?'
  PULL NLIGNES
  DO I=1 TO NLIGNES
    "FETCH C1"
```

```
      IF SQLSTATE¬='00000' THEN DO; CALL ERR; LEAVE; END
      SAY 'LINE ' I '-------------------------:'
      DO J=1 TO $COLNT.0
         A='COLUMN ' J '=' COL.J
         IF NULL_COL.J THEN A=A " --(NULL)--"
         SAY A
      END
   END
 "CLOSE C1"
   IF SQLSTATE¬='00000' THEN CALL ERR
   SAY 'GIVE YOUR SELECT'
   PULL SELECT
   END
 "DISC"
 RETURN
 ERR: SAY 'ERROR-----------------------'
      SAY 'SQLSTATE=' SQLSTATE
      SAY 'SQLCODE=' SQLCODE
      IF SQLSTATE='02000' THEN SAY '(END OF DATA)'
      RETURN
```

ERROR MESSAGES

The following error messages might occur:

- $IRX015E NUMBER OF ARGUMENTS INVALID

  (Self explanatory.)

- $IRX045E DB2: SYNTAX ERROR

  A syntax error has been detected in an SQL clause, or the cursor name is not C1, C2, H1, or H2. Notice that REXXPLUS/MVS detects syntax errors only in the parts of the clause that are of interest to it; in a DECLARE CURSOR, for example, REXXPLUS/MVS examines the syntax of ‘DECLARE xx CURSOR <WITH HOLD> FOR ’, but a syntax error in the following SELECT clause will be detected by DB2, not by REXXPLUS/MVS.

- $IRX046E STRING IS TOO LONG

  The SQL clause is too long.

- $IRX047E FETCH WITHOUT PREVIOUS DECLARE

(Self explanatory.)

- $IRX048E ERROR WHEN STORING IN REXX VARIABLE

  (Self explanatory.) The most frequent reason is lack of memory.

- $IRX049E RETURN CODE OF EXEC SQL ¬=0

  The return code of an internal DB2 instruction ('EXEC SQL EXECUTE IMMEDIATE' or 'EXEC SQL PREPARE xxx INTO :SQLDA') is not zero. This return code is R15, not the SQLCODE. The most frequent reason for such an error is that the connection with DB2 has not been established. In this case, DB2 can't fill up the SQLCODE.

  You must always test the return code from the clause 'CONN', returned in the RC and REASON variables. You must not proceed if RC is not zero.

- $IRX050E $DB2INST: AN ARGUMENT IS INVALID

  (Self explanatory.)

- $IRX054E $DB2INST FUNCTION NOT PREVIOUSLY USED

  Before executing any SQL clause, you must execute the $DB2INST function once.

PROGRAMMING NOTES

The following should be noted:

- WORKEXT_USERFIELD

  $IRX is the main table in REXXPLUS/MVS. This table is pointed to by WORKEXT_USERFIELD, in the REXX work block extension. When I wrote REXXPLUS/MVS, some years ago, the Name/Token service (MVS SP422 and later) did not exist, but now it's available and offers a standard way of controlling an anchor point. Nevertheless, my method works.

- Floating point data

You know it is not obvious how to convert an extended decimal number with a large exponent (eg12.3E+25) to an internal binary floating point number, and *vice versa*, so I used the $IRXCNV2 COBOL program.

$IRXCNV2

```
ID DIVISION.
       PROGRAM-ID. IRXCNV2.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       LINKAGE SECTION.
       01  FONCTION PIC 9(2) COMP.
*******    0: CONV DOUBLE FLOAT->EXTENDED
*******    4: CONV EXTENDED->DOUBLE FLOAT
*******    8: CONV FLOAT->EXTENDED
*******   12: CONV EXTENDED->FLOAT
       01  FLOT8  COMP-2.
       01  ZONE   REDEFINES FLOT8.
         02 FLOT4 COMP-1.
         02 FILLER PIC X(4).
       01  ETENDU PIC +V.9999999999999999E+99.
       PROCEDURE DIVISION USING FONCTION, FLOT8, ETENDU.
           IF      FONCTION =  0 THEN MOVE FLOT8 TO ETENDU
           ELSE IF FONCTION =  4 THEN MOVE ETENDU TO FLOT8
           ELSE IF FONCTION =  8 THEN MOVE FLOT4 TO ETENDU
           ELSE IF FONCTION = 12 THEN MOVE ETENDU TO FLOT4.
           GOBACK.
```

Editor's note: the code will be continued in the next issue of DB2 Update.

Patrick Leloup
System Engineer
Credit Agricole de Loire Atlantique (France)

Code published in *DB2 Update* is available from our Web site, www.xephon.com. Once you have registered, you can select an article that you want e-mailed to you. Remember to have your copy of the issue containing the article with you when you access our Web site.

Data generator for DB2 – part 4

This month we conclude the code for a tool that will generate any kind of test data (with any degree of complexity).

EINZDEF1

```
PANEL EINZDEF1

)ATTR
+ TYPE(OUTPUT) COLOR(&CSØNITC) INTENS(LOW)  SKIP(ON)
_ TYPE(OUTPUT) COLOR(&CSØNITC) INTENS(LOW)  SKIP(ON) JUST(RIGHT)
% TYPE(TEXT)   COLOR(&CSØHITC) INTENS(HIGH) SKIP(ON)
¬ TYPE(INPUT)  COLOR(&CSØNIIF) INTENS(LOW)  PAD('_') CAPS(ON)
FORMAT(MIX)
# TYPE(INPUT)  COLOR(&CSØNIIF) INTENS(LOW)  PAD(' ')
? TYPE(INPUT)  COLOR(&CSØNIIF) INTENS(LOW)  PAD('Ø') JUST(RIGHT)
! TYPE(TEXT)                   INTENS(LOW)  SKIP(ON)
)BODY EXPAND(··)
!· · Field-definition· ·
% COMMAND ===>#ZCMD
%
! S = edit , U = Switch to unique or back
! |
! v      Sub-                                          Field    already
defined
! S Nr.  Typ    Field name              Datatyp         Length   V
Unique or not
! ------------------------------------------------------------------V----
-------
)MODEL
 ¬Z_NR%  _Z%   +FELD               %  +DATATYP  %     _LANG %  +Z+UNI
!
)INIT
.ZVARS = 'CHOICE ST DEF'
.AUTOSEL = YES
)PROC
 VER(&CHOICE LIST,S,U)
)END
```

FEHLDB2

```
PANEL FEHLDB2

)ATTR
```

```
! TYPE(FP)
$ TYPE(LI)  PADC(USER) CAPS(OFF)
% TYPE(NEF) PADC(USER) CAPS(ON )
# AREA(SCRL)
)BODY WINDOW(5Ø,4) CMD(ZCMD)
!Command ===>%Z                                  !
#SAREA39                                           #
#                                                  #
#                                                  #
)AREA SAREA39
!$etext1                              !
!$etext2                              !
)INIT
.ZVARS = '(ZCMD)'
&ZWINTTL = 'DB2 Table Error'
)END
```

INTEDEF1

```
PANEL INTEDEF1

)ATTR
! TYPE(FP)
$ TYPE(NEF) CAPS(ON)            JUST(RIGHT)
% TYPE(LI)  CAPS(ON) PADC(USER)
+ TYPE(NEF) CAPS(ON) PADC(USER)
# AREA(SCRL)
)BODY WINDOW(5Ø,8) CMD(ZCMD)
!Command ===>+Z                                  !
#SAREA39                                           #
#                                                  #
#                                                  #
#                                                  #
#                                                  #
#                                                  #
)AREA SAREA39
!Field %FELD                !

!Startvalue:$SIN        !Endvalue:$EIN        !

!Stepvalue :       $IIN ! (optional)
)INIT
.ZVARS = '(ZCMD)'
.CURSOR = SIN
&ZWINTTL = 'INTEGER-Field Definition'
)PROC
VER (&SIN,RANGE,-2147483648,2147483647)
VER (&EIN,RANGE,-2147483648,2147483647)
VER (&IIN,RANGE,Ø,9999)
)END
```

MESS000

```
Change M_pref to the M_pref you defined in REXX DATAMAI2
  and create a member M_pref.ØØ in your ISPMLIB

M_prefØØØ  'Datatype required' .ALARM = YES
'Possible datatypes: CHAR, INTEGER, SMALLINT, DECIMAL, TIMESTAMP, DATE
or TIME'

M_prefØØ1  'Length missing or wrong' .ALARM = YES
'Max. length for CHAR = Ø..254 '

M_prefØØ2  'Sub-Type only for CHAR' .ALARM = YES
'SUBDEF-split only valid for CHAR and VARCHAR.'

M_prefØØ3  'No or wrong main entry' .ALARM = YES
'SUBDEF-split only valid for CHAR and VARCHAR.'

M_prefØØ4  'Length SUBDEFs>main entry' .ALARM = YES
'The sum of all SUBDEF length is greater than the length of the main
entry.'

M_prefØØ5  'DELETE need a Nr' .ALARM = YES
'A DELETE need a valid Nr. and perhaps a Sub-Typ Nr.'

M_prefØØ6  'Entry not found' .ALARM = YES
'DELETE only possible with valid Nr.'

M_prefØØ7  'Input required' .ALARM = YES
'Input is required !'

M_prefØØ8  'Entry exists!' .ALARM = YES
'There is an entry with the same Nr. (Sub-Typ Nr.)'

M_prefØØ9  'No valid structure' .ALARM = YES
'&Messtext only works with a defined structure.'
```

MESS001

```
Change M_pref to the M_pref you defined in REXX DATAMAI2
  and create a member M_pref.Ø1 in your ISPMLIB

M_prefØ1Ø  '&E_Y no leap year' .ALARM = YES
'The year &E_Y was/is not a leap year, therefore 29.Ø2. not possible.'

M_prefØ12  'Start-date > End-date' .ALARM = YES
'Start-date must be less than End-date. Sorry.'

M_prefØ13  'Start-value > End-value' .ALARM = YES
```

```
'Start-value must be less than End-value. Sorry.'

M_pref014  'Step > Range' .ALARM = YES
'The interval is greater than the difference between Start-/Endvalue.'

M_pref015  'Def. not possible' .ALARM = YES
'The definition of a main entry is not possible, if a SUBDEF exist.'

M_pref016  'Still &proz % left ?' .ALARM = YES
'What should I do with the remaining &proz percent ???'

M_pref017  '&proz % to much' .ALARM = YES
'Percent is from 100. Now we have &proz percent too much !!!'

M_pref018  'No structure avail.' .ALARM = YES
'There is no structure available. Define first, please !'

M_pref019 'Nothing saved !' .ALARM = YES
'The definition wasn't saved, because date not valid -> NO LEAP YEAR !'
```

MESS002

```
Change M_pref to the M_pref you defined in REXX DATAMAI2
  and create a member M_pref.02 in your ISPMLIB

M_pref020  'No SUBDEF possible' .ALARM = YES
'It is not possible to define a Sub-Type, because the main entry is
still def.'

M_pref021  'DSN is missing' .ALARM = YES
'To save a definition I need a dataset-name. '

M_pref022  'DSN not catalogued ' .ALARM = YES
'&checkdsn is not in the catalog.'

M_pref023  'DSN will be overwritten' .ALARM = YES
'&checkdsn will be overwritten.'

M_pref024  'MEMBER-name required' .ALARM = YES
'&checkdsn is a PO-dataset, please specify member-name.'

M_pref025  'Dataset too small' .ALARM = YES
'A min. of &bytes bytes is required for DSN &checkdsn'

M_pref026  '2 times this DSN' .ALARM = YES
'DSN for generation and output must be different.'

M_pref027  '&iostext' .ALARM = YES
'&iotext'
```

```
M_pref028  'End < Start-Date' .ALARM = YES
'DATE not valid -> End-date < Start-date not possible.'

M_pref029  'End < Start-Time' .ALARM = YES
'TIME not valid -> End-time < Start-time not possible.'
```

RDDB2DEF

```
/*----REXX RDDB2DEF - Select the structure of a given table ---------*/
 parse arg creator tabelle ok
 db2ssid = 'db2_ssid'              /* Put your DB2-subsystem ID here */
 seltab  = "TEMP"                                  /* Do not change */
 selparm = 'SELECT COLNO * 2 AS COLNO, NAME, COLTYPE, ',
           'LENGTH AS LANG FROM SYSIBM.SYSCOLUMNS',
           "WHERE TBNAME = '"tabelle"' AND TBCREATOR ='"creator"'",
           'ORDER BY 1';
 "ISPEXEC VPUT (SELPARM SELTAB DB2SSID) SHARED"
 "ISPEXEC SELECT PGM(DB2ISPF)"
return
```

SMALDEF1

```
PANEL SMALDEF1

)ATTR
! TYPE(FP)
$ TYPE(NEF) CAPS(ON)            JUST(RIGHT)
% TYPE(LI)  CAPS(ON) PADC(USER)
+ TYPE(NEF) CAPS(ON) PADC(USER)
# AREA(SCRL)
)BODY WINDOW(50,8) CMD(ZCMD)
!Command ===>+Z                                  !
#SAREA39                                          #
#                                                 #
#                                                 #
#                                                 #
#                                                 #
#                                                 #
)AREA SAREA39
 Field %FELD                  !

!Startvalue : $SSI   !   Endvalue : $ESI   !

!Stepvalue  :    $ISI! (optional)
)INIT
.ZVARS = '(ZCMD)'
.CURSOR = SSI
```

```
&ZWINTTL = 'SMALLINT-Field definition'
)PROC
VER (&SSI,RANGE,-32768,32767)
VER (&ESI,RANGE,-32768,32767)
VER (&ISI,RANGE,Ø,999)
)END
```

STRUDEF1

```
PANEL STRUDEF1  (change M_pref to your prefix)

)ATTR
 + TYPE(OUTPUT) COLOR(&CSØNITC) INTENS(LOW)  SKIP(ON) CAPS(OFF)
 _ TYPE(OUTPUT) COLOR(&CSØNITC) INTENS(LOW)  SKIP(ON) JUST(RIGHT)
 % TYPE(TEXT)   COLOR(&CSØHITC) INTENS(HIGH) SKIP(ON)
 ¬ TYPE(INPUT)  COLOR(&CSØNIIF) INTENS(LOW)  PAD('_') CAPS(ON)
FORMAT(MIX)
 # TYPE(INPUT)  COLOR(&CSØNIIF) INTENS(LOW)  PAD(' ') JUST(RIGHT)
 ? TYPE(INPUT)  COLOR(&CSØNIIF) INTENS(LOW)  PAD('Ø') JUST(RIGHT)
 $ TYPE(INPUT)  COLOR(&CSØNIIF) INTENS(LOW)  PAD(' ')
 ! TYPE(TEXT)                   INTENS(LOW)  SKIP(ON)
)BODY EXPAND(··)
!· ·General Structure Definition· ·
% Command ===>$ZCMD
%
! A = Add ; D = Delete
! |
! V      Sub-                                          Field-
! S Nr.  Typ    Field-name              Data-Typ       Length    already
defined
!¬Z?NT%  #Z%   ¬FELT              %    ¬DATATYT  %    #LANT!     V
Unique or not
! ----------------------------------------------------------------V----
-------
)MODEL
   _NR%  _Z%   +FELD                %  +DATATYP  %    _LANG !   +Z+UNI
!
)INIT
.ZVARS = 'CHOICE SD ST DEF'
 IF (&CHOICE = ' ') , .CURSOR=CHOICE
)PROC
 IF (&CHOICE = ' ') , .CURSOR=CHOICE

 IF (&CHOICE = 'A')
   IF (&DATATYT ¬= 'DECIMAL')
     IF (&NT     ¬= ' ')
       IF (&FELT  = ' ')    , .MSG    = M_prefØØ7 , .CURSOR=FELT
     IF (&FELT    = ' ')    , .CURSOR=FELT
     IF (&FELT   ¬= ' ')
```

```
      IF (&DATATYT = ' ')  , .MSG    = M_pref000 , .CURSOR = DATENTYT
     IF (&DATATYT = 'CHAR') , VER (&LANT  NB,RANGE,1,254,MSG=DDF001)
     IF (&DATATYT = 'VARCHAR') , IF (&LANT  = ' ') , .MSG    = M_pref001
     IF (&DATATYT ¬= 'CHAR')
       IF (&DATATYT ¬= 'VARCHAR')
         IF (&SD    ¬= ' ') , .MSG    = M_pref002 , .CURSOR = SD
     IF (&DATATYT = 'DECIMAL')   , IF (&LANT  = ' ') , .MSG = M_pref001
     IF (&DATATYT = 'SMALLINT')  , &LANT = '2'
     IF (&DATATYT = 'INTEGER')   , &LANT = '11'
     IF (&DATATYT = 'TIMESTAMP') , &LANT = '26'
     IF (&DATATYT = 'TIME')      , &LANT = '8'
     IF (&DATATYT = 'DATE')      , &LANT = '10'

 IF (&CHOICE = 'D')
   IF (&NT   = ' ') , .MSG = M_pref005

 VER(&CHOICE LIST,A,D)
 VER(&DATATYT LIST,INTEGER,SMALLINT,CHAR,DECIMAL,TIMESTAMP,DATE,TIME, +
             VARCHAR)
 VER(&NT NUM) , VER(&SD NUM) , VER(&LANT NUM)
)END
```

STRUDEF2

```
PANEL STRUDEF2

)ATTR
! TYPE(FP)
$ TYPE(NEF)  PADC(USER) CAPS(ON)
# AREA(SCRL)
)BODY WINDOW(40,4) CMD(ZCMD)
!Command ===>$Z                         !
#SAREA39                                 #
#                                        #
#                                        #
)AREA SAREA39
!   Entry will be deleted ?$Z!(y/n)

)INIT
.ZVARS = '(ZCMD STRUDEL)'
.CURSOR = STRUDEL
&ZWINTTL = 'Confirm delete'
)PROC
VER (&STRUDEL,LIST,Y,N)
)END
```

STRUDEF3

```
PANEL STRUDEF3
```

```
)ATTR
! TYPE(FP)
$ TYPE(NEF)  PADC(USER) CAPS(ON)
# AREA(SCRL)
)BODY WINDOW(3Ø,4) CMD(ZCMD)
!Command ===>$Z              !
#SAREA39                     #
#                            #
#                            #
)AREA SAREA39
 !Precision :$PT!Scale :$SC!

)INIT
.ZVARS = '(ZCMD)'
.CURSOR = PT
IF (&PT = ' ') , &PT = '5' , &SC = 'Ø'
&ZWINTTL = 'DECIMAL Definition'
)PROC
VER (&PT RANGE,1,18) , VER (&SC RANGE,Ø,&PT)
)END
```

TIMEDEF1

```
PANEL TIMEDEF1

)ATTR
! TYPE(FP)
$ TYPE(NEF) PADC('Ø')  CAPS(ON) JUST(RIGHT)
% TYPE(NEF) PADC(USER) CAPS(ON)
+ TYPE(LI)  PADC(USER) CAPS(ON)
# AREA(SCRL)
)BODY WINDOW(5Ø,8) CMD(ZCMD)
!Command ===>%Z                                    !
#SAREA39                                            #
#                                                   #
#                                                   #
#                                                   #
#                                                   #
#                                                   #
)AREA SAREA39
!Field +FELD                  !

!Starttime :$Z $Z $Z ! Endtime :$Z $Z $Z !

!Interval  :$Z !     Step Unit :%Z ! (HH, MM, SS)
)INIT
.ZVARS = '(ZCMD THS TMS TSS THE TME TSE TIT TEI)'
.CURSOR = THS
```

```
&ZWINTTL = 'TIME-Field definition'
)PROC
VER (&THS,RANGE,ØØ,24) , VER (&TMS,RANGE,ØØ,6Ø) , VER (&TSS,RANGE,ØØ,6Ø)
VER (&THE,RANGE,ØØ,24) , VER (&TME,RANGE,ØØ,6Ø) , VER (&TSE,RANGE,ØØ,6Ø)
VER (&TEI,LIST,SS,MM,HH)
IF (&THE = 24)   , &TME = ØØ , &TSE = ØØ
IF (&THS = 24)   , &TMS = ØØ , &TSS = ØØ
IF (&TEI = SS)
  IF (&TIT = 6Ø) , &TEI = 'MM' , &TIT = 'Ø1'
IF (&TEI = MM)
  IF (&TIT = 6Ø) , &TEI = 'HH' , &TIT = 'Ø1'
IF (&TEI = HH)   , VER (&TIT,RANGE,ØØ,24)
IF (&TEI = MM)   , VER (&TIT,RANGE,ØØ,6Ø)
IF (&TEI = SS)   , VER (&TIT,RANGE,ØØ,6Ø)
IF (&TEI = '  ')
  IF (&TIT ¬= '  ') , &TEI = 'SS'
)END
```

TISTDEF1

```
PANEL TISTDEF1

)ATTR
! TYPE(FP)
$ TYPE(NEF) CAPS(ON)            JUST(RIGHT)
% TYPE(LI)  CAPS(ON) PADC(USER)
+ TYPE(NEF) CAPS(ON) PADC(USER)
# AREA(SCRL)
)BODY WINDOW(44,1Ø) CMD(ZCMD)
!Command ===>+Z                              !
#SAREA39                                      #
#                                             #
#                                             #
#                                             #
#                                             #
#                                             #
#                                             #
#                                             #
)AREA SAREA39
  !Field %FELD                  !

              yyyy-mm-dd-hh.mm.ss.mmmmmm
  !Starttime:$xj  $xm$xt$xh$xi$xs$xms   !
  !Endtime  :$yj  $ym$yt$yh$yi$ys$yms   !
  !Interval :$zi    !Step Unit :$zs!    !
   Valid Step Units: YY,MO,DD,HH,MI,SS,MS
)INIT
.ZVARS = '(ZCMD)'
.CURSOR = XJ
```

```
&ZWINTTL = 'TIMESTAMP-Field definition'
)PROC
VER (&XJ RANGE,0001,9999) , VER (&XM RANGE,1,12) , VER (&XT RANGE,1,31)

IF (&XM='4'  OR &XM='6' OR &XM='9' OR &XM='11')
  VER (&XT,RANGE,1,30)
IF (&XM='2') , VER (&XT,RANGE,1,29)

VER (&XH  RANGE,00,24) , VER (&XI  RANGE,00,59) , VER (&XS  RANGE,00,59)
VER (&XMS RANGE,000000,999999)
IF (&YJ ¬= '    ')
  IF (&XJ = '    ') , &XJ = 0001

VER (&YJ RANGE,&XJ,9999) , VER (&YM RANGE,1,12) ,  VER (&YT RANGE,1,31)

IF (&YM='4'  OR &YM='6' OR &YM='9' OR &YM='11')
  VER (&YT,RANGE,1,30)
IF (&YM='2')
  VER (&YT,RANGE,1,29)

VER (&YH  RANGE,00,24) , VER (&YI  RANGE,00,59) , VER (&YS  RANGE,00,59)
VER (&YMS RANGE,000000,999999)
IF (&ZI ¬= '  ') , IF (&ZS = '  ') , &ZS = 'SS'
IF (&ZS ¬= 'MS')
  IF (&ZI > 99)  , &ZI = 99
  VER (&ZI RANGE,00,99)
IF (&XH = '24')
  &XI  = '00' , &XS  = '00' , &XMS = '000000'
IF (&YH = '24')
  &YI  = '00' , &YS  = '00' , &YMS = '000000'
VER (&ZS LIST,YY,MO,DD,HH,MM,SS,MS)
)END
```

Rainer Cockx
Systems Programmer
R+V Versicherung AG (Germany)

If you would like to contribute an article to *DB2 Update*, a copy of the *Notes for contributors* can now be downloaded from our Web site. Point your browser at www.xephon.com/contnote.html.

XDB Systems has announced Version 2.0 of its ExpressLane data access middleware, providing connectivity between PC-based graphical environments and mainframe databases including DB2, IMS, and VSAM.

The new version includes an ODBC Static Bind capability and a Catalog Cache facility. The former pre-packages database calls and binds them to mainframe databases, so making them go faster. With the Catalog Cache, requests to the DB2 system catalog can be resolved on an NT-based server, bypassing the mainframe altogether.

For further information contact:
XDB Systems, 9861 Broken Land Parkway, Columbia, MD 21046, USA.
Tel: (410) 312 9300.
XDB (UK) Ltd, Wessex House, 80 Park Street, Camberley, Surrey, GU15 3PT, UK.
Tel: (01276) 686662.

* * *

VMark Software has announced Release 3.0 of its DataStage data extraction and transformation tool. New features include change data capture, mainframe data access, and a new set of developer productivity tools.

The change data capture facility is said to avoid complete rebuilds of large datasets by updating only the data that's been changed since a specified date or event.

Meanwhile, the new version provides access to virtually all legacy data sources, regardless of platform, including VSAM, IMS, and DB2 systems via plug-in stages which read the legacy files directly into DataStage's metadata repository.

For further information contact:
VMark Software, 50 Washington Street, Westboro, MA 01581-1021, USA.
Tel: (508) 366 3888.
VMark Software, Edenfield, London Road, Bracknell, Berks, RG12 2XH, UK.
Tel: (01344) 355500.

* * *

USoft has launched Version 4.2 of its USoft Developer business rules-based development tool, which has more functionality and more configuration alternatives such as native support for DB2.

Other features include an easier to use TeamWork with a better import wizard, plus a new wizard to help users store new data and documents in the repository. There's also better memory management and performance, USoft batch performance improvements, more options in control processing, and extended XA support.

For further information contact:
Usoft, 1000 Marina Boulevard, Brisbane, CA 94005, USA.
Tel: (415) 875 3300.