134

DB2

December 2003

update

In this issue

DB2 Update

Published by

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

North American office

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

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 1999 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/db2; 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 £100 ($160) per 1000 words and £50 ($80) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £20 ($32) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

A procedure that generates and executes a QUIESCE statement for one or more databases

The QUIESCE online utility establishes a quiesce point for a tablespace, partition, tablespace set, or list of table spaces and table space sets, and records it in the SYSIBM.SYSCOPY catalog table. A successful QUIESCE improves the probability of a successful RECOVER or COPY. You should run QUIESCE frequently between regular executions of COPY to establish regular recovery points for future point-in-time recovery.

In our shop we add a QUIESCE step before every batch program that updates DB2 data. In order to establish one complete QUIESCE point (all tablespaces in a database) I developed a procedure that generates and executes a QUIESCE statement for all tablespaces in a given database. The procedure accepts a wildcard in the databasename.

JCL QUIESCE

```
//QUIESCE  EXEC DB2QSCE,
//             JOBNAME='jobname',
//             DBNAME='dbname'
```

JCL PROCEDURE DB2QSCE

```
//DB2QSCE  PROC DB2S='DB2B',      ** DB2-SUBSYSTEM
//             HLQSAM='',         ** HIGH-LEVEL QUALIFIER WORK-FILE
//             JOBNAME='',        ** JOB NAME
//             DBNAME=''          ** DB2 DATABASENAME
//*****************************************************
//***      ALLOCATE WORK-FILES
//*****************************************************
//STEPØ1Ø  EXEC PGM=IEFBR14
//SORTOUT  DD DSN=&HLQSAM..&JOBNAME..QUIESCE,
//            DISP=(MOD,DELETE),
//            UNIT=WORK,SPACE=(TRK,(1,1))
//*****************************************************
//***      GENERERATE QUIESCE-STATEMENT
//*****************************************************
//STEPØ2Ø  EXEC PGM=IKJEFTØ1,DYNAMNBR=2Ø,
//         PARM=(QUIESCE,'&DBNAME')
```

```
//SYSEXEC  DD DISP=SHR,DSN=REXX-LIBRARY
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//QUIESCE  DD DSN=&HLQSAM..&JOBNAAM..QUIESCE,
//            DISP=(NEW,CATLG),
//            UNIT=WORK,
//            SPACE=(CYL,(1,1),RLSE)
//SYSTSIN  DD DUMMY
//SYSUDUMP DD SYSOUT=*
//*****************************************************
//***      EXECUTE QUIESCE STATEMENT
//*****************************************************
//STEPØ3Ø  EXEC PGM=DSNUTILB,PARM='&DB2S,&JOBNAME..QUIESCE'
//SYSIN    DD DISP=SHR,DSN=&HLQSAM..&JOBNAME..QUIESCE
//SYSPRINT DD  SYSOUT=*
//UTPRINT  DD  SYSOUT=*
//SORTOUT  DD  DISP=(NEW,PASS),UNIT=WORK,SPACE=(CYL,(1,1Ø),RLSE)
//SYSUT1   DD  DISP=(NEW,PASS),UNIT=WORK,SPACE=(CYL,(1,1Ø),RLSE)
//SYSERR   DD  DISP=(NEW,PASS),UNIT=WORK,SPACE=(CYL,(1,1Ø),RLSE)
//*****************************************************
//***      DELETE WORK-FILES
//*****************************************************
//STEP999  EXEC PGM=IEFBR14
//SORTOUT  DD DSN=&HLQSAM..&JOBNAME..QUIESCE,
//            DISP=(MOD,DELETE),
//            UNIT=WORK,SPACE=(TRK,(1,1))
```

REXX PROGRAM QUIESCE

```
/* Rexx ************************************************************ */
/* Name     : QUIESCE                                                */
/* Date     : 28-5-2ØØ3                                              */
/* Author   : Loet Polkamp                                           */
/* Function : Generate QUIESCE statement for all tablespaces in the  */
/*            selected database.                                     */
/* Execute  : Batch (using the DSNREXX DB2-interface)                */
/* Input    : Parameter: databasename, wildcard possible (*)         */
/* Output   : SYSIN-dataset                                          */
/* Checks   : rc=1ØØ1: no parm specified                             */
/*            rc=1ØØ2: database not found                            */
/*            rc=1ØØ3: unknown DB2 subsystem                         */
/* ***************************************************************** */
Trace o
Arg param
If param='' then do
   erc =1ØØ1
   emsg='No databasename specified'
   call REXX_error
   End
```

```
search=translate(param,'%','*')
/* ************* S T A R T   M A I N L I N E ********************* */
Call Init
Call Dbamvars
Call ConnDB2
Call Get_spaces
Call DiscDB2
Call Write_stmt
Call Exit
/* ************ S T A R T   P R O C E D U R E S ***************** */
Init:
  t=0 /* fetched rows */
Return
DBAMVARS:   /* Get DB2 subsystem & loadlib from SYSIN-member "DSNTEP" */
  "ALLOC FI(INPUT) DA('SYSINLIB(DSNTEP)') SHR REUSE"
  'execio 1 diskr INPUT (stem input. finis)'
  "FREE FI(INPUT)"
  parse var input.1 run program plan lib line
  x=pos("'",lib) + 1
  lib=substr(lib,x,20)
  x=pos("'",lib) - 1
  lib=substr(lib,1,x)
  DBAVDB2L = lib
  "ALLOC FI(INPUT) DA('CENE.PROCLIB(DB2CAFIN)') SHR REUSE"
  'execio 1 diskr INPUT (stem input. finis)'
  "FREE FI(INPUT)"
  parse var input.1 db2id line
  DBAVDB2S=DB2ID
  Select
     when (dbavdb2s='DB2A') then dbavsyst='PROD'
     when (dbavdb2s='DB2B') then dbavsyst='TEST'
     when (dbavdb2s='DB2C') then dbavsyst='SPIN'
     when (dbavdb2s='DB2D') then dbavsyst='ACPT'
     otherwise do
        erc =1003
        emsg='Unknown DB2 subsystem'
        call REXX_error
        End
  End
Return
Get_spaces:
  SQL_stmt="SELECT DBNAME,NAME FROM SYSIBM.SYSTABLESPACE
            WHERE DBNAME LIKE '"search"%' ORDER BY NAME"
  ADDRESS DSNREXX "EXECSQL PREPARE S1 FROM :SQL_stmt"
  erc=sqlcode; esigl=sigl; emsg=sqlerrmc
  If erc<>0 then Call REXX_error
  ADDRESS DSNREXX "EXECSQL OPEN C1"
  erc=sqlcode; esigl=sigl; emsg=sqlerrmc
  If erc<>0 then Call REXX_error
  Do until erc=100
```

```
     ADDRESS DSNREXX "EXECSQL FETCH C1 INTO :dbname,:tsname"
     erc=sqlcode; esigl=sigl; emsg=sqlerrmc
     Select
        when erc=0  then do
             t=t+1
             tsname.t=tsname
             dbname.t=dbname
             End
        when erc=100 & t=0 then do
             erc =1002
             emsg='Databasename 'dbname' does not exist'
             call REXX_error
             End
        when erc=100 & t>0 then nop
        Otherwise Call REXX_error
     End
  End
  ADDRESS DSNREXX "EXECSQL CLOSE C1 "
  erc=sqlcode; esigl=sigl; emsg=sqlerrmc
  If erc<>0 then Call REXX_error
  ADDRESS DSNREXX "EXECSQL COMMIT"
  erc=sqlcode; esigl=sigl; emsg=sqlerrmc
  If erc<>0 then Call REXX_error
Return erc
Write_stmt:
  Do x=1 to t
        If x=1 then Queue '  QUIESCE TABLESPACE 'dbname.x'.'tsname.x
               else Queue '          TABLESPACE 'dbname.x'.'tsname.x
        "EXECIO 1 DISKW QUIESCE";
     End
  If rc=0 then Say 'QUIESCE-statement generated for DB2-database 'search
Return
ConnDB2: /* Connect to DB2          */
  'SUBCOM DSNREXX'  /* Is host command env avbl ? */
  If rc then        /* If not then add it         */
     S_rc = RXSUBCOM('ADD','DSNREXX','DSNREXX')
     ADDRESS DSNREXX
     ADDRESS DSNREXX "CONNECT" dbavdb2s
Return
DiscDB2: /* Disconnect from DB2          */
     ADDRESS DSNREXX "DISCONNECT"
Return
Rexx_error:
  Say '*************************************************'
  Say '* ERROR-message                    Progr: SBHRQSCE'
  Say '* ----------------------------------------------'
  Say '* Return code: 'erc
  Say '* Message    : 'emsg
  Say '*************************************************'
  Exit erc
```

```
Exit:
  Exit
/* *************** P R O G R A M      E N D ********************** */
```


Automating DB2 utilities handling

At our installation we are using IBM DB2 utilities and also BMC utilities (COPY PLUS, LOAD PLUS, RECOVER PLUS, etc). The BMC utilities were handled through the BMC command processor (BMCDSN, started task BMCXABU in our case).

Handling DB2 utilities this way means it is necessary to know the syntax of the commands and the prefix for each of the DB2 utilities (IBM -DSNP, BMC +DSNP, in our case), which is not very practical.

To make it easier to handle the DB2 utilities I developed the program DB2UTI, which allows the centralized handling of utilities without users needing to know the syntax/prefix of the commands.

DB2UTI (REXX/ISPF) shows in the main panel all the DB2 utilities (IBM and BMC) and their status (active/stopped/paused/ ...). The utilities are activated by means of options.

The program has been used by operators and programmers (in the development and test environments), which frees support staff from this task.

The main options are **D** to display details of a DB2 utility (in a window, including a utility command) and **T** to terminate the utility. If the utility is in active status, the program shows a confirmation window before termination.

The program has help panels (F1 key). In addition, it has explanatory and error messages in two versions – short and long

(accessible by pushing the F1 key when there is a query).

It maintains a log in which it writes all the users' actions. It stores the option chosen (D, T, ' ' to display the main panel), which user executes DB2UTI, the DB2 subsystem, the status of the utility, the utility-id, the phase of the utility, which user executes the utility, whether it's an IBM or BMC utility, and the date and time of DB2UTI execution.

At the start of its execution the program searches the DB2 subsystems defined in OS/390 (read the DB2 vectors from storage), which are then shown in an initial panel from where the user can choose the subsystem to be used.

DB2UTI is a REXX/ISPF program, which uses tables and ISPF windows. At present the program is executed with OS/390 V2.9, DB2 V6.1, and the BMC command processor BMCDSN V2.R3.00, which is as an option on the panel DB2 DSNEPRI (beside SPUFI, QMF).

The library 'your.xxxx.windows.user' is dsorg PO, lrecl 40, blksize 4000, and the library 'your.xxxx.windowsj.user' is dsorg PO, lrecl 50, blksize 5000.

DB2UTIXE

```
/* REXX */
TRACE OFF
NUMERIC   DIGITS 12;
CVT     = C2X(STORAGE(10,4))                  /* ADDR DE CVT          */
CVTJESCT= D2X((X2D(CVT))+296)                 /* POINTER A CVTJESCT   */
JESCT   = C2X(STORAGE(CVTJESCT,4))            /* ADDR DE JESCT        */
JESSSCT = D2X((X2D(JESCT))+24)                /* POINTER A JESSSCT    */
SSCVT   = C2X(STORAGE(JESSSCT,4))             /* ADDR DE SSCVT        */
J = 0
DO I = 1  WHILE (SSCVT <> 00000000)
          SSCTSNAM=D2X((X2D(SSCVT))+8)        /* POINTER A SSCTSNAM   */
          ERLY  = D2X((X2D(SSCVT))+20)        /* POINTER A ERLY       */
          ERLYAD= C2X(STORAGE(ERLY,4))        /* ADDR DE ERLY         */
          ERLYSCOM= D2X((X2D(ERLYAD))+56)     /* POINTER A ERLYSCOM   */
          IF SUBSTR(STORAGE(ERLYSCOM,64),29,8) = 'DSN3EPX ' THEN
             DO
             J = J + 1
             SSIDDB2.J = STORAGE(SSCTSNAM,4) /* ADDR DE SSCTSNAM     */
```

```
           END
         SSCTSCTA = D2X((X2D(SSCVT))+4)     /* POINTER AL SGTE SSCVT*/
         SSCVT    = C2X(STORAGE(SSCTSCTA,4))
END
"ISPEXEC LIBDEF ISPPLIB DATASET ID('your.PANELI')"
"ISPEXEC LIBDEF ISPMLIB DATASET ID('your.MSGLBI')"
"ISPEXEC TBCREATE TSSIDDB2",
"NAMES(O SDB2)",
"NOWRITE REPLACE"
O= '';
DO J = 1 TO J
SDB2 = SSIDDB2.J
"ISPEXEC TBADD   TSSIDDB2"
END
"ISPEXEC TBTOP   TSSIDDB2"
"ISPEXEC TBDISPL TSSIDDB2 PANEL(DB2RESSP)"
IF RC = 8 THEN EXIT;
"ISPEXEC LIBDEF ISPPLIB"
"ISPEXEC LIBDEF ISPMLIB"
"ISPEXEC TBEND  TSSIDDB2"
SSID = SDB2
SELECT
  WHEN SSID = 'DB2P' THEN
       DO
      "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.PROD.PANELLIB.USER')"
      "ISPEXEC LIBDEF ISPMLIB DATASET ID('YOUR.PROD.MSGLIB')"
       BMCABU = 'BMCXABU'
       LOGPREFIX = 'PROD'
       END
  WHEN SSID = 'DB2D' THEN
       DO
      "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.DESA.PANELLIB.USER')"
      "ISPEXEC LIBDEF ISPMLIB DATASET ID('YOUR.DESA.MSGLIB')"
       BMCABU = 'BMCDABU'
       LOGPREFIX = 'DESA'
       END
  WHEN SSID = 'DB2T' THEN
       DO
      "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.TEST.PANELLIB.USER')"
      "ISPEXEC LIBDEF ISPMLIB DATASET ID('YOUR.TEST.MSGLIB')"
       BMCABU = 'BMCTABU'
       LOGPREFIX = 'TEST'
       END
  OTHERWISE
       DO
      "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.PROD.PANELLIB.USER')"
      "ISPEXEC LIBDEF ISPMLIB DATASET ID('YOUR.PROD.MSGLIB')"
       BMCABU = 'BMCXABU'
       LOGPREFIX = 'PROD'
       END
```

```
END
"ISPEXEC TBCREATE TUTI",
"NAMES(O STATUS UTILID UTILITY PHASE UTIUSER UTISOFT ",
"DBNAME SPNAME STARTT COMMAND)",
"NOWRITE REPLACE"
CALL BMCABUCHK                    /* BMCABU PARA TERM UTILITARIOS BMC */
DO FOREVER
O= '';
CALL DISUTI;
CALL DISUTIBMC;
IF NOHAYUTIIBM = 1 & NOHAYUTIBMC = 1 THEN
   DO
   ADDRESS ISPEXEC "SETMSG MSG(DBC2Ø1)"
   CALL GRABALOG
   END
ELSE
   DO
  "ISPEXEC TBSORT  TUTI FIELDS(STATUS,C,D,UTILID,C,A,UTILITY,C,A)"
  "ISPEXEC TBTOP   TUTI"
   END
NOHAYUTIIBM = Ø
NOHAYUTIBMC = Ø
TERUTIIBM = Ø
TERUTIBMC = Ø
"ISPEXEC TBDISPL TUTI PANEL(DB2UTIP)"
IF   RC = 8 THEN
     DO
    "ISPEXEC LIBDEF ISPPLIB"
    "ISPEXEC LIBDEF ISPMLIB"
    "ISPEXEC TBEND  TUTI"
     EXIT
     END
ELSE DO
        IF PFKEYIN = 'S' THEN "ISPEXEC DISPLAY PANEL(DB2UTIH)";
        IF ZTDSELS  > Ø THEN
           DO
           CALL TRATA_UTILITARIO;
           END;
        DO WHILE ZTDSELS  > 1
          "ISPEXEC TBDISPL TUTI"
           IF ZTDSELS > Ø THEN
              DO
              CALL TRATA_UTILITARIO;
              END
        END
        O = ''
        CALL TBDELROWS
     END
END
EXIT
```

```
/*------------------- R U T I N A S --------------------------*/
BMCABUCHK:
X = OUTTRAP(JST.)
INTERPRET "STATUS  "BMCABU;
X = OUTTRAP('OFF')
BMCABUOK = 'N'
DO I = 1 TO JST.Ø
IF WORD(JST.I,1) = 'IKJ56211I' THEN BMCABUOK = 'S'     /* EXECUTING */
END
SELECT
  WHEN BMCABUOK = 'S' THEN NOP
  OTHERWISE DO
            ADDRESS ISPEXEC "SETMSG MSG(DBC2Ø5)"
           "ISPEXEC TBDISPL TUTI PANEL(DB2UTIP)"
            CALL GRABALOG
            EXIT
            END
END
RETURN
/*-------------------------------------------------------------*/
DISUTI:
W = OUTTRAP('UTI.')
    QUEUE "-DIS UTILITY(*)"
    QUEUE "END"
   "DSN SYSTEM("SSID")"
W = OUTTRAP('OFF')
    IF RC > Ø THEN
       DO
       ADDRESS ISPEXEC "SETMSG MSG(DBC2ØØ)"          /* NO HAY DB2 */
       RETURN
       END
IF SUBSTR(UTI.1,1,8) = 'DSNU112I' THEN    /* NO HAY UTILITARIOS DB2 */
   DO
   NOHAYUTIIBM  = 1;
   IF TERUTIIBM = 1 THEN ADDRESS ISPEXEC "SETMSG MSG(DBC2Ø4)"
   IF TERUTIBMC = 1 THEN ADDRESS ISPEXEC "SETMSG MSG(DBC2Ø7)"
   RETURN
   END
I = 1;
DO I = I WHILE I < UTI.Ø
   IF (SUBSTR(UTI.I,1,8) = 'DSNU1Ø5I' |,         /* UTILITY ACTIVE  */
       SUBSTR(UTI.I,1,8) = 'DSNU1ØØI' ) THEN     /* UTILITY STOPPED */
      DO J = 1 TO 7
         SELECT
         WHEN J = 1  THEN UTIUSER = WORD(UTI.I,7)
         WHEN J = 2  THEN NOP
         WHEN J = 3  THEN UTILID  = WORD(UTI.I,3)
         WHEN J = 4  THEN NOP
         WHEN J = 5  THEN UTILITY = WORD(UTI.I,3)
         WHEN J = 6  THEN PHASE   = WORD(UTI.I,3)
```

```
         WHEN J = 7  THEN STATUS  = WORD(UTI.I,3)
         END
         IF J < 7 THEN I = I + 1;
      END
   IF (SUBSTR(UTI.I,1,8) = 'DSNU1Ø6I' ) THEN /* UTILITY TERMINATING */
      DO J = 1 TO 3
         SELECT
         WHEN J = 1  THEN UTILITY = WORD(UTI.I,5)
         WHEN J = 2  THEN NOP
         WHEN J = 3  THEN UTILID  = WORD(UTI.I,3)
         END
         UTIUSER = ''
         PHASE   = ''
         STATUS  = 'TERMTING'
         IF J < 3 THEN I = I + 1;
      END
      UTISOFT = 'IBM'
     "ISPEXEC TBADD TUTI"
      DISUTI_ADD = 1
      CALL GRABALOG
END
RETURN
/*-----------------------------------------------------------------*/
DISUTIBMC:
CALL ALLOCSYS
 SYSIN.1  = "SELECT * "
 SYSIN.2  = "FROM BMCUTIL.CMN_BMCUTIL;"
"EXECIO  *  DISKW SYSIN     (STEM SYSIN."
"EXECIO  Ø  DISKW SYSIN     (FINIS"
 ADDRESS TSO "ALLOC FILE(SYSIN)"    "DATASET('"DSNIN"')    OLD REUSE"
 ADDRESS TSO "ALLOC FILE(SYSRECØØ)" "DATASET('"DSNREC"')   OLD REUSE"
 ADDRESS TSO "ALLOC FILE(SYSPRINT)" "DATASET('"DSNPRINT"') OLD REUSE"
 ADDRESS TSO "ALLOC FILE(SYSPUNCH) DUMMY"
 PUSH "END"
 PUSH "RUN PROGRAM(DSNTIAUL) PLAN(DSNTIAUL) PARMS('SQL')"
 ADDRESS TSO "DSN SYSTEM("SSID")"
 ADDRESS TSO "EXECIO * DISKR SYSRECØØ (STEM BMCUTI. FINIS"
 ADDRESS TSO "FREE FILE(SYSIN)"
 ADDRESS TSO "FREE FILE(SYSRECØØ)"
 ADDRESS TSO "FREE FILE(SYSPRINT)"
 ADDRESS TSO "FREE FILE(SYSPUNCH)"
 CALL DELTMP
 UTISOFT = 'BMC'
 IF BMCUTI.Ø = Ø THEN                    /* NO HAY UTILITARIOS BMC */
    DO
    NOHAYUTIBMC  = 1;
    IF TERUTIIBM = 1 THEN ADDRESS ISPEXEC "SETMSG MSG(DBC2Ø4)"
    IF TERUTIBMC = 1 THEN ADDRESS ISPEXEC "SETMSG MSG(DBC2Ø7)"
    RETURN
    END
```

```
 DO I = 1 TO BMCUTI.Ø
    UTILID   = SUBSTR(BMCUTI.I,1,16)
    STATUS   = SUBSTR(BMCUTI.I,17,1)
    IF STATUS = 'X' THEN STATUS = 'ACTIVE'
    IF STATUS = 'S' THEN STATUS = 'STOPPED'
    IF STATUS = 'A' THEN STATUS = 'ACTIVE'
    IF STATUS = 'P' THEN STATUS = 'PAUSED'
    UTILITY  = SUBSTR(BMCUTI.I,18,8)
    PHASE    = SUBSTR(BMCUTI.I,26,8)
    UTIUSER  = SUBSTR(BMCUTI.I,34,8)
    SSID     = SUBSTR(BMCUTI.I,42,4)
    DBNAME   = SUBSTR(BMCUTI.I,55,8)
    SPNAME   = SUBSTR(BMCUTI.I,63,8)
    STARTT   = RIGHT(BMCUTI.I,26)
    COMMAND  = SUBSTR(BMCUTI.I,8Ø,256)      /* COMMAND VARCHAR(256) */
 /* SAY STATUS UTILID UTILITY PHASE UTIUSER UTISOFT DBNAME SPNAME   */
   "ISPEXEC TBADD TUTI";
    DISUTI_ADD = 1
    CALL GRABALOG
 END
RETURN
/*-------------------------------------------------------------------*/
DISUTIBMCDET:
CMD1 = SUBSTR(COMMAND,ØØ1,34)
CMD2 = SUBSTR(COMMAND,Ø35,34)
CMD3 = SUBSTR(COMMAND,Ø69,34)
CMD4 = SUBSTR(COMMAND,1Ø3,34)
CMD5 = SUBSTR(COMMAND,137,34)
CMD6 = SUBSTR(COMMAND,171,34)
CMD7 = SUBSTR(COMMAND,2Ø5,34)
CMD8 = SUBSTR(COMMAND,239,18)
"ISPEXEC ADDPOP POPLOC(O)"
SELECT
  WHEN SSID = 'DB2P' THEN
 "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.PROD.WINDOWSJ.USER')"
  WHEN SSID = 'DB2D' THEN
 "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.DESA.WINDOWSJ.USER')"
  WHEN SSID = 'DB2T' THEN
 "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.TEST.WINDOWSJ.USER')"
  OTHERWISE
 "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.PROD.WINDOWSJ.USER')"
END
"ISPEXEC DISPLAY PANEL(DB2TUTDJ)"
"ISPEXEC REMPOP"
SELECT
  WHEN SSID = 'DB2P' THEN
 "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.PROD.PANELLIB.USER')"
  WHEN SSID = 'DB2D' THEN
 "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.DESA.PANELLIB.USER')"
  WHEN SSID = 'DB2T' THEN
```

```
 "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.TEST.PANELLIB.USER')"
  OTHERWISE
 "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.PROD.PANELLIB.USER')"
END
RETURN
/*-------------------------------------------------------------*/
CONFIRM:
C = ''
"ISPEXEC ADDPOP POPLOC(O)"
SELECT
  WHEN SSID = 'DB2P' THEN
 "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.PROD.WINDOWS.USER')"
  WHEN SSID = 'DB2D' THEN
 "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.DESA.WINDOWS.USER')"
  WHEN SSID = 'DB2T' THEN
 "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.TEST.WINDOWS.USER')"
  OTHERWISE
 "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.PROD.WINDOWS.USER')"
END
"ISPEXEC DISPLAY PANEL(DB2TUTIC)"
IF C = 'S' THEN CONFIRMRC = Ø
ELSE CONFIRMRC = 1;
"ISPEXEC REMPOP"
SELECT
  WHEN SSID = 'DB2P' THEN
 "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.PROD.PANELLIB.USER')"
  WHEN SSID = 'DB2D' THEN
 "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.DESA.PANELLIB.USER')"
  WHEN SSID = 'DB2T' THEN
 "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.TEST.PANELLIB.USER')"
  OTHERWISE
 "ISPEXEC LIBDEF ISPPLIB DATASET ID('YOUR.PROD.PANELLIB.USER')"
END
RETURN
/*-------------------------------------------------------------*/
TERUTI:
IF UTISOFT = 'IBM' THEN
   DO
   W = OUTTRAP('TER.')
       QUEUE "-TERM UTILITY("UTILID")"
       QUEUE "END"
      "DSN SYSTEM("SSID")"
       TERRC = RC
   W = OUTTRAP('OFF')
   IF TERRC = Ø THEN
      DO
      ADDRESS ISPEXEC "SETMSG MSG(DBC2Ø4)"       /*   OK TERM UTIL DB2*/
      TERUTIIBM = 1;
      CALL GRABALOG;
      END
```

```
   ELSE
      ADDRESS ISPEXEC "SETMSG MSG(DBC2Ø3)"       /*FALLO TERM UTIL DB2*/
   END
IF UTISOFT = 'BMC' THEN
   DO
   DSNZT    = USERID() || '.UTI' || '.IN'  || TIME('S')
      ADDRESS TSO "ALLOC FILE(ZTEMPN)  DATASET('"DSNZT"') " ,
               "NEW CAT REUSE UNIT(SYSDA)" ,
               "LRECL(8Ø) BLKSIZE(2792Ø) RECFM(F B) SPACE(1,1) CYL"
   PLANNM  = 'ABUD23ØØ'
   OPTNAME = ''
   QUEUE '+TERM UTIL('UTILID')'
   QUEUE 'END'
   BMCDSN = 'ABUDSN /'SSID'/'PLANNM'/'OPTNAME'/'
   X = OUTTRAP("OUTPUT.",'*',"NOCONCAT")
   BMCDSN
   IF (RC = Ø) THEN
      DO
      ADDRESS ISPEXEC "SETMSG MSG(DBC2Ø7)"       /* OK   TERM UTIL BMC*/
      TERUTIBMC = 1;
      CALL GRABALOG;
      END
   ELSE
      ADDRESS ISPEXEC "SETMSG MSG(DBC2Ø6)";      /*FALLO TERM UTIL BMC*/
   "EXECIO" OUTPUT.Ø "DISKW" ZTEMPN "(OPEN STEM OUTPUT.)"
   IF RC > Ø THEN SAY 'WRITE RC = ' RC
   "EXECIO"        Ø "DISKW" ZTEMPN "(STEM OUTPUT FINIS)"
   IF RC > Ø THEN SAY 'CLOSE RC = ' RC
   Y = OUTTRAP(OFF)
   IF TERUTIBMC = 1 THEN
      DO
     "FREE DDNAME(ZTEMPN)"
      W = OUTTRAP(DELZT.)
      ADDRESS TSO "DELETE ('"DSNZT"')"
      W = OUTTRAP('OFF')
      END
   END
RETURN
/*-------------------------------------------------------------*/
TBDELROWS:
"ISPEXEC TBSTATS TUTI ROWCURR(TUTIROWS)"
"ISPEXEC TBTOP TUTI"
DO I = 1 TO TUTIROWS
"ISPEXEC TBSKIP   TUTI "
"ISPEXEC TBDELETE TUTI"
END
RETURN
/*-------------------------------------------------------------*/
ALLOCSYS:
DSNIN    = USERID() || '.UTI' || '.IN'  || TIME('S')
```

```
DSNREC   = USERID() || '.UTI' || '.REC' || TIME('S')
DSNPRINT = USERID() || '.UTI' || '.PRT' || TIME('S')
   ADDRESS TSO "ALLOC FILE(SYSIN)   DATASET('"DSNIN"') " ,
            "NEW CAT REUSE UNIT(SYSDA)" ,
            "LRECL(8Ø) BLKSIZE(2792Ø) RECFM(F B) SPACE(1,1) CYL"
            IF RC¬= Ø THEN
               DO
                 ADDRESS ISPEXEC "SETMSG MSG(DBCØØ1)"
                 ALLOCSYSRC = 1
                 RETURN
               END
/*-------------------------------------------------------------*/
   ADDRESS TSO "ALLOC FILE(SYSRECØØ)  DATASET('"DSNREC"') " ,
            "NEW CAT REUSE UNIT(SYSDA)" ,
            "LRECL(1ØØ) BLKSIZE(27ØØ) RECFM(F B) SPACE(1,1) CYL"
            IF RC¬= Ø THEN
               DO
                 ADDRESS ISPEXEC "SETMSG MSG(DBCØØ2)"
                 ALLOCSYSRC = 1
                 RETURN
               END
/*-------------------------------------------------------------*/
   ADDRESS TSO "ALLOC FILE(SYSPRINT)   DATASET('"DSNPRINT"') " ,
            "NEW CAT REUSE UNIT(SYSDA)" ,
            "LRECL(133) BLKSIZE(2793Ø) RECFM(F B) SPACE(1,1) CYL"
            IF RC¬= Ø THEN
               DO
                 ADDRESS ISPEXEC "SETMSG MSG(DBCØØ3)"
                 ALLOCSYSRC = 1
                 RETURN
               END
RETURN
/*-------------------------------------------------------------*/
GRABALOG:
ADDRESS TSO "ALLOC FILE(LOG) DSNAME('"LOGPREFIX".DB2UTI.LOG') OLD REUSE"
IF RC = Ø THEN
   DO
     ADDRESS TSO "EXECIO * DISKR LOG (STEM LOG. FINIS"
     IF RC = Ø THEN ULTIMO = LOG.Ø + 1
   END
ELSE
   ADDRESS TSO "ALLOC FILE(LOG) DSNAME('"LOGPREFIX".DB2UTI.LOG')" ,
            "NEW CAT REUSE UNIT(SYSDA)" ,
            "LRECL(8Ø) BLKSIZE(2792Ø) RECFM(F B) SPACE(1,1) CYL"
            IF RC <> Ø THEN
               DO
                 ADDRESS ISPEXEC "SETMSG MSG(DBCØ41)"
                 RETURN
               END
USERIDL = LEFT(USERID(),6)
```

```
SSIDL   = LEFT(SSID,4)
SELECT
  WHEN (NOHAYUTIIBM = 1 & NOHAYUTIBMC = 1) THEN
    LOG.ULTIMO = ' ' || USERIDL   || SSIDL   || LEFT(' ',8),
                     || LEFT(' ',17)         || LEFT(' ',8),
                     || LEFT(' ',8)          || LEFT(' ',3),
                     || DATE('S') || TIME()
  WHEN DISUTI_ADD  = 1 THEN
    LOG.ULTIMO = ' ' || USERIDL   || SSIDL   || LEFT(STATUS,8),
                     || LEFT(UTILID,17)      || LEFT(PHASE,8),
                     || LEFT(UTIUSER,8)      || LEFT(UTISOFT,3),
                     || DATE('S') || TIME()
  OTHERWISE
    LOG.ULTIMO =  0  || USERIDL   || SSIDL   || LEFT(STATUS,8),
                     || LEFT(UTILID,17)      || LEFT(PHASE,8),
                     || LEFT(UTIUSER,8)      || LEFT(UTISOFT,3),
                     || DATE('S') || TIME()
END
ADDRESS TSO "EXECIO  *  DISKW LOG  (STEM LOG."
ADDRESS TSO "EXECIO  Ø  DISKW LOG  (FINIS"
"FREE DDNAME(LOG)"
DISUTI_ADD  = Ø
RETURN
/*-----------------------------------------------------------------*/
DELTMP:
W = OUTTRAP(DELTMP.)
ADDRESS TSO "DELETE ('"DSNIN"')"
ADDRESS TSO "DELETE ('"DSNREC"')"
ADDRESS TSO "DELETE ('"DSNPRINT"')"
W = OUTTRAP('OFF')
RETURN
/*-----------------------------------------------------------------*/
TRATA_UTILITARIO:
/*SAY STATUS UTILID UTILITY PHASE UTIUSER UTISOFT;        */
CONFIRMRC = Ø
IF O = 'T' THEN
   SELECT
        WHEN STATUS = 'ACTIVE' & PHASE <> ' ' THEN
             DO
                                              CALL CONFIRM
             IF CONFIRMRC = Ø                THEN CALL TERUTI ;
             END
        WHEN STATUS = 'ACTIVE' & PHASE =  ' ' & UTILITY = 'COPY' THEN
             DO
                                              CALL CONFIRM
             IF CONFIRMRC = Ø                THEN CALL TERUTI ;
             END
        WHEN STATUS = 'ACTIVE' & PHASE =  ' ' THEN CALL TERUTI;
        WHEN STATUS = 'STOPPED'               THEN CALL TERUTI;
        /* STATUS = ' '*/
```

```
         WHEN UTILITY = 'RECOVER' & UTISOFT = 'BMC' THEN
              DO
              POSP = POS('.',UTILID)
              IF POSP > Ø THEN
                 DO
                 IF SUBSTR(UTILID,1,POSP-1) = UTIUSER THEN
                    DO
                    JOBNAME = STRIP(SUBSTR(UTILID,POSP+1,18-POSP))
                    W = OUTTRAP(JOB.)
                   "STATUS "JOBNAME;
                    W = OUTTRAP('OFF')
                    JOBMAX = JOB.Ø
/*JOB ACTIVO*/      IF SUBSTR(JOB.JOBMAX,1,9) = 'IKJ56211I' THEN
                       DO
                       /* SAY JOBNAME ' ACTIVO'    */
                                                     CALL CONFIRM
                       IF CONFIRMRC = Ø        THEN CALL TERUTI ;
                       END
                    ELSE
                       DO
                       /* SAY JOBNAME ' NO ACTIVO' */
                                                     CALL TERUTI ;
                       END
                    END
                 END
              END
         OTHERWISE
              DO
                                                     CALL CONFIRM
              IF CONFIRMRC = Ø              THEN CALL TERUTI ;
              END
   END
IF O = 'D' THEN
   DO
   IF UTISOFT = 'BMC' THEN
      DO
      CALL DISUTIBMCDET;
      CALL GRABALOG;
      END
   ELSE
      DO
      ADDRESS ISPEXEC "SETMSG MSG(DBC2Ø8)"
      END
   END
RETURN
/* -----------------  FIN  END  ---------------------------------*/
```

MEMBER YOUR.PANELI(DB2RESSP)

```
)ATTR
```

```
 % TYPE(TEXT)   INTENS(HIGH) SKIP(ON)
   color(turquoise)
 + TYPE(TEXT)   INTENS(LOW)  SKIP(ON)
 _ TYPE(INPUT)  INTENS(HIGH) CAPS(ON)            HILITE(USCORE)
 ! TYPE(OUTPUT) INTENS(LOW)  CAPS(ON) JUST(LEFT) HILITE(REVERSE)
   color(turquoise)
 @ TYPE(TEXT)   INTENS(HIGH) CAPS(ON)            HILITE(REVERSE)
   color(turquoise)
)BODY EXPAND(\\)
@                            SUBSYSTEM(S) DB2
@
%COMMAND ===>_ZCMD
%
%                        Opc  Subsystem  DB2
+                       %--  --------------
)MODEL
+                        _O+ !SDB2+
)INIT
 .help   = db2ressh
 .cursor = O
 &SCRO   = PAGE
)REINIT
 .cursor = O
)PROC
if (.PFKEY = 'PF01')
    &pfkeyin = 's'
IF (&ZTDSELS > 0000)
    VER(&O,LIST,S)
)END
```

MEMBER YOUR.PANELI(DB2RESSH)

```
)attr
 % TYPE(TEXT)   INTENS(HIGH) SKIP(ON)
   color(turquoise)
 + TYPE(TEXT)   INTENS(LOW)  SKIP(ON)
 _ TYPE(INPUT)  INTENS(HIGH) CAPS(ON)            HILITE(USCORE)
 ! TYPE(OUTPUT) INTENS(LOW)  CAPS(ON) JUST(LEFT) HILITE(REVERSE)
   color(turquoise)
 @ TYPE(TEXT)   INTENS(HIGH) CAPS(ON)            HILITE(REVERSE)
   color(turquoise)
)BODY EXPAND(\\)
@                            SUBSYSTEM(S) DB2
@
%   Opc (OPTIONS) valids :+
+
+  %S+        - To select the subsystem DB2.
+
+
%  %Description of the columns:+
```

```
+
+  %Subsystem + Name/Identification of subsystem DB2
+               defined in MVS - OS/390 (SYS1.PARMLIB).
+
)END
```

MEMBER YOUR.xxxx.PANELLIB.USER(DB2UTIP)

```
)ATTR
 % TYPE(TEXT)   INTENS(HIGH) SKIP(ON)
   color(turquoise)
 + TYPE(TEXT)   INTENS(LOW)  SKIP(ON)
 _ TYPE(INPUT)  INTENS(HIGH) CAPS(ON)            HILITE(USCORE)
 ! TYPE(OUTPUT) INTENS(LOW)  CAPS(ON) JUST(LEFT) HILITE(REVERSE)
   color(turquoise)
 @ TYPE(TEXT)   INTENS(HIGH) CAPS(ON)            HILITE(REVERSE)
   color(turquoise)
)BODY EXPAND(\\)
@                        UTILITIES  DB2  (IBM/BMC)
@
%COMMAND ===>_ZCMD                                    +%SCROLL ===>_SCRO+
%
%
%   Opc  Status    Utilid             Utility   Phase     User
Software
+  %--  --------  ---------------  --------  --------  --------  ------
--
)MODEL
+   _O+ !STATUS  +!UTILID          +!UTILITY +!PHASE   +!UTIUSER
+!UTISOFT+
)INIT
 .help   = db2utih
 .cursor = o
 &SCRO   = PAGE
)REINIT
 .cursor = o
)PROC
if (.PFKEY = 'PF01')
    &pfkeyin = 's'
IF (&ZTDSELS > 0000)
    VER(&O,LIST,T,D)
)END
```

MEMBER YOUR.xxxx.PANELLIB.USER(DB2UTIH)

```
)attr
 % TYPE(TEXT)   INTENS(HIGH) SKIP(ON)
   color(turquoise)
 + TYPE(TEXT)   INTENS(LOW)  SKIP(ON)
```

```
 _ TYPE(INPUT)  INTENS(HIGH) CAPS(ON)           HILITE(USCORE)
 ! TYPE(OUTPUT) INTENS(LOW)  CAPS(ON) JUST(LEFT) HILITE(REVERSE)
   color(turquoise)
 @ TYPE(TEXT)   INTENS(HIGH) CAPS(ON)           HILITE(REVERSE)
   color(turquoise)
)BODY EXPAND(\\)
@                        UTILITIES  DB2  (IBM/BMC)
@
%
%   Opc (OPTIONS) :
%
+   T        - Terminate a utility DB2   (IBM or BMC).
%
+   D        - Display detail of a utility DB2 (BMC).
%
%   Description of the columns:
%
+   Status   - Status of the utility (ACTIVE,STOPPED,PAUSED,etc)
%
+   Utilid   - Identifier/Name of the utility.
%
+   Utility  - Utility (COPY/UNLOAD/LOAD/RUNSTAT/REORG,etc).
%
+   Phase    - Phase of the utility at the time DB2UTI executes.
%
+   User     - User who executes the utility.
%
+   Software - Manufacturer of the utility, may be IBM or BMC.
%
)INIT
)END
```

MEMBER YOUR.XXXX.MSGLIB(DBC00)

```
DBC001   'FAILURE OF SYSIN ALLOC'    .ALARM=YES
'CHECK FOR OUT OF SPACE IN SYSDA/RACF PERMISSIONS'
DBC002   'FAILURE OF SYSREC ALLOC'   .ALARM=YES
'CHECK FOR OUT OF SPACE IN SYSDA/RACF PERMISSIONS'
DBC003   'FAILURE OF SYSPRINT ALLOC' .ALARM=YES
'CHECK FOR OUT OF SPACE IN SYSDA/RACF PERMISSIONS'
```

MEMBER YOUR.XXXX.MSGLIB(DBC04)

```
DBC041   'REXX LOG NOT ACCESSIBLE' .ALARM=YES
'FAILURE IN ACCESS/DEFINE THE REXX LOG. (MBVS.DB2COPIA.LOG)'
```

MEMBER YOUR.XXXX.MSGLIB(DBC20)

```
DBC200   'CONNECTION DB2 FAULT'     .ALARM=YES  .WINDOW=LNORESP
'THE CONNECTION DB2 IS NOT AVAILABLE. STARTUP THE DB2 SUBSYSTEM'
DBC201   'UTILITIES DB2 NOT FOUND'  .ALARM=YES  .WINDOW=LNORESP
'NOT FOUND UTILITIES DB2 IBM/BMC (ACTIVE/STOP)'
DBC203   'TERM UTIL IBM FAULT'      .ALARM=YES  .WINDOW=LNORESP
'FAILURE IN COMMAND TERM UTIL DB2 (NOT BMC). CALL TO SUPPORT'
DBC204   'UTILITY DB2 IBM FINISHED' .ALARM=YES .WINDOW=LNORESP
'THE UTILITY DB2 IBM WAS FINISHED SUCCESSFULLY'
DBC205   'STARTUP BMC_ABU'          .ALARM=YES  .WINDOW=LNORESP
'BMC COMMAND PROCESSOR NOT AVAILABLE. STARTUP BMC_ABU'
DBC206   'TERM UTIL BMC FAULT'      .ALARM=YES  .WINDOW=LNORESP
'FAILURE IN COMMAND TERM UTIL BMC (NOT IBM). CALL TO SUPPORT'
DBC207   'UTILITY BMC FINISHED'     .ALARM=YES .WINDOW=LNORESP
'THE UTILITY DB2 BMC WAS FINISHED SUCCESSFULLY'
DBC208   'DETAILS VALID FOR BMC'    .ALARM=YES .WINDOW=LNORESP
'THE DISPLAY OF DETAILS IS VALID ONLY FOR BMC UTILITIES'
```

MEMBER YOUR.XXXX.WINDOWSJ.USER(DB2TUTDJ)

(lrecl 50 blksize 5000):

```
)ATTR
 % TYPE(TEXT)   INTENS(LOW)  SKIP(ON)
                   COLOR(TURQUOISE)
 @ TYPE(TEXT)   INTENS(HIGH) CAPS(ON)
   HILITE(REVERSE) COLOR(TURQUOISE)
 & TYPE(TEXT)   INTENS(HIGH) CAPS(ON)
   HILITE(REVERSE) COLOR(TURQUOISE)
 $ TYPE(TEXT)   INTENS(HIGH) CAPS(ON)
   HILITE(REVERSE) COLOR(BLUE)
 + TYPE(OUTPUT) INTENS(HIGH) CAPS(ON)
   HILITE(REVERSE) COLOR(RED)
 ! TYPE(OUTPUT) INTENS(HIGH) CAPS(ON)
   HILITE(REVERSE) COLOR(TURQUOISE)
)BODY WINDOW(50,16)
$   DETAIL OF  UTILID  !UTILID            $
$-----------------------------------------------
%
%  $ DATABASE%!DBNAME  %
%  $SPACENAME%!SPNAME  %
%  $   INICIO%!STARTT                    %
%
%  $  COMANDO%!CMD1
%             !CMD2
%             !CMD3
%             !CMD4
%             !CMD5
```

```
%             !CMD6
%             !CMD7
%             !CMD8
)INIT
)PROC
)END
```

MEMBER YOUR.XXXX.WINDOWS.USER(DB2TUTIC)

(lrecl 40 blksize 4000):

```
)attr
 % TYPE(TEXT)   INTENS(HIGH) SKIP(ON)
 + TYPE(TEXT)   INTENS(LOW)  SKIP(ON)
   HILITE(REVERSE) color(blue)
 _ TYPE(INPUT)  INTENS(HIGH) CAPS(ON)
   HILITE(REVERSE) color(RED)
 ! TYPE(OUTPUT) INTENS(HIGH) CAPS(ON)
   HILITE(REVERSE) color(turquoise)
 @ TYPE(TEXT)   INTENS(HIGH) SKIP(OFF)
   HILITE(REVERSE) color(turquoise)
)BODY WINDOW(40,10) ASIS
@     CONFIRMATION OF TERM UTILITY
@
@
@ +CONFIRMATION OF TERM UTILITY@
@ +      !UTILID           +==>_C@(S|N)
@
@
@
@
@
)INIT
 .CURSOR = C
)PROC
VER(&C,NONBLANK,LIST,S,N)
)END
```

Carlos German Osorio Montoya
Database Administrator (Peru)

Accessing DB2 using a Web browser and DB2/REXX

When we think of connecting from a PC to DB2 running on a mainframe, we generally think of DB2 Connect, ODBC, JDBC – a lot of stuff to set-up, and often a difficult learning process. This article explores getting to mainframe DB2 using something we all have on our PC – a Web browser such as Internet Explorer – and mainframe DB2/REXX. There are some additional mainframe requirements, which are described below.

MAINFRAME REQUIREMENTS

On the mainframe, you need to have a Web server running – this will serve the Web pages. It is known as the http server or the Websphere http server. The started task is normally called IMWEBSRV. The http server itself requires Unix System Services (USS), previously called OpenEdition. You need to be set up as a user in USS, or you need an OMVS RACF segment and a home directory set up.

THE WEB SERVER

The function of a Web server is to deliver Web pages to its clients – the people connected to it using a Web browser. The pages consist of HTML. The server will also send any graphics that are referred to in the HTML. The browser and server need to use a common protocol so that they understand each other – this is the Hypertext Transport Protocol, or http.

There are two basic ways that the server will construct the pages it's going to send: the first is by reading a piece of HTML stored in a file somewhere; the second is by running a program that writes the HTML to standard output. We're interested in the latter, programmatic, method.

The traditional programmatic method is the Common Gateway

Interface (CGI). CGI programs are typically written in a scripting language such as Perl or a compiled language such as C.

More recently, Java solutions such as Java servlets, Java Beans, and Enterprise Java Beans are being favoured. These offer many built-in services such as session persistence and security, and an object library which has useful facilities for the developer. There are large software and set-up costs for this, though – WebSphere Application Server, an additional-cost item, is required to run the Java Virtual Machine on the mainframe and maintain the environment in which the servlets and beans run. In addition, there's a lot to learn – Java, JDBC or SQLJ, and the tools needed to build the directory structure and maintain it (this is complex to do manually).

For our simple light method, we'll use CGI. Remember though, that although these two programmatic approaches seem so different, they are essentially doing the same thing – writing out pages of HTML.

The most popular Web server in the world is probably Apache, the open source Web server from the Unix world. The mainframe http server is based on Apache, although there are a number of differences.

The Web server has a large configuration file, which may run to several thousand lines. This contains a lot of directives controlling how the server works – we need to be familiar with only a few of them.

URLS

It's important to understand the concept of a URL (Universal Resource Locator). This is the means the Web browser uses to locate a particular resource. You'll be familiar with URLs through typing them into the browser address box, eg http://www.thingie.com/docs/test.html.

The URL consists of these components:

- http:// – the scheme. For our purposes, we're just interested in http.

- www.thingie.com – the host name. This is a name which is resolved by a Domain Name Server (DNS) to an actual IP address.
- /docs/ – the directory path name.
- test.html – the file name within the directory path.

It's worth looking at the directory path a bit more closely. It looks like we have a directory called **docs** at the root level on the www.thingie.com server. This is probably not the case.

The Web server's configuration file has a directive called DocumentRoot. This specifies the 'logical' root for the server. The directory path name in the URL is relative to the document root. For instance, if the document root is */usr/local/apache/htdocs*, then our URL above refers to */usr/local/apache/htdocs/docs/test.html*. If no file name is given, a default file name is searched for: this is usually index.html. The default is specified by the directive DirectoryIndex.

Let's look at another example URL:

http://www.thingie.com/cgi-bin/testprog.sh?name=Alan+Smith.

This example is calling a CGI program. The program name is *testprog.sh*. The directory path, in this case ***/cgi-bin/***, is arrived at differently for a CGI program.

When configuring the server, you have the choice of letting a CGI program be executed from anywhere – you denote that it's CGI by giving it a suffix such as .cgi – or you specify that CGI programs can be executed only from a specific directory hierarchy or hierarchies. The former method is generally seen as being a security risk, so the latter is general practice. The ScriptAlias directive specifies the valid path for CGI programs in the configuration file. In our example, there would be a ScriptAlias entry for /cgi-bin/.

Lastly, a query string follows the question mark in the URL. The query string consists of the parameters that are passed to the CGI program. These relate to form fields that were filled in on the

HTML page that generated the URL. Each parameter is of the form field=value. In the example, the field is name and the value is Alan+Smith. If there were several fields, they would be separated by ampersands (&). On the Web page, the contents of the name field were 'Alan Smith'. The space has been transformed into a plus sign (+). Other special characters also get transformed. Obviously, plus signs and ampersands need to be transformed because, if they are left in the query string, they get interpreted as spaces and parameter delimiters respectively. Such characters are represented in the string as a per cent sign (%), followed by the hex representation of the ASCII code for the character. For instance, a plus sign is encoded as %2B, an ampersand is encoded as %26. Of course, the per cent sign has to be encoded if that occurs in a field – it is %25. The CGI program that receives the query string needs to parse and translate the query string to retrieve the individual field values.

CGI

The Web server runs a CGI program in response to a request from a Web browser. Often this is caused by the submission of a form on a Web page. The HTML for the form to send information to the URL in the previous section might look like this:

```
<FORM METHOD="GET" ACTION="http://www.thingie.com/cgi-bin/testprog.sh">
<p>Enter your name: <INPUT TYPE=TEXT NAME=subsys>
</FORM>
```

The job of the CGI program is to retrieve any information it needs to perform its processing, then write its output, in the form of HTML, to standard output. This response is returned to the user's Web browser.

The Web server passes information to the CGI program by placing it in environment variables. Environment variables are a Unix construct and are like global variables. A calling process can set them; processes it calls can then look at the values of the variables.

If the submitted form has a method of GET, the form's field values are passed as part of the URL, as we saw in the previous section.

This query string is placed in the environment variable QUERY_STRING.

If the form has a method of POST, the query string is placed in the program's standard input and the program has to read it from there. In both cases, the string has to be parsed to retrieve the field values.

When it comes to writing its output, the program has to output a header followed by the HTML to create the Web page at the user end. The header is very simple – it consists of this line:

```
Content-type: text/html
```

followed by a blank line (which denotes the end of the header).

THE WEB SERVER ON THE MAINFRAME

The WebSphere http server is based on Apache, although there are some differences in the configuration file.

The configuration file can be several thousand lines, but there are only a few things in it that you need to be familiar with. The file is a file, not a dataset, ie it's held in Unix System Services.

There are two ways of getting into USS. ISHELL presents a panel-based interface, which should be relatively easy to use for ISPF users. OMVS looks more like a traditional Unix shell or command line interface. Let's look at the configuration file using ISHELL.

Type **TSO ISHELL** to get in. You should see something like this:

```
File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
------------------------------------------------------------------------
                            OpenMVS ISPF Shell

Enter a pathname and do one of these:

    - Press Enter.
    - Select an action bar choice.
    - Specify an action code or command on the command line.

Return to this panel to work with a different pathname.
```

```
                                                                More:     +
   /u/smithac
   _____________________________________________________________________
   _____________________________________________________________________
   _____________________________________________________________________
```

The current working directory is shown as */u/smithac*. This is my home directory. Change this to the directory containing the configuration file, */etc/* and press *Enter*. The display should then look something like this:

```
                                Directory List

/etc/
Select one or more files with / or action codes.

  Type  Filename
_ Dir   .
_ Dir   ..
_ File  auto.master
_ File  banner
_ Dir   booksrv
_ Dir   bpa
_ File  cics.map
_ File  cicsservice.map
_ Dir   cmx
_ File  DB2.map
_ File  DB2service.map
_ Dir   dce
_ Dir   dfs
_ Syml  hosts
_ File  httpd.conf
_ File  httpd.conf.D14Oct2002
```

Put a **b** against the file httpd.conf to browse it.

Earlier we covered the Apache directives, which specify the server's document root and the location of CGI programs. These are different in the WebSphere http server.

With Apache, you specify the document root with the DocumentRoot directive. With WebSphere, you use the **Pass** directive to specify how directory paths in the URL are mapped to actual directories in Unix System Services. Consider these two directives:

```
Pass            /DB2/*              /Web/Webdept/nudbs/*
Pass            /*                  /Web/pub/*
```

The first specifies that, if the directory path in the URL is */DB2/* , the actual directory on the server used is */Web/Webdept/nudbs/* . So the URL http://www.things.co.uk/DB2/tests/test1.html would map to a real file location */Web/Webdept/nudbs/tests/test1.html*.

The second directive specifies the location of the document root. Note that earlier directives are matched first.

The specification of CGI program directories works in the same way. The **Exec** directive performs the same function as **Pass** for directories containing CGI programs, eg:

```
Exec              /bookmgr-cgi/*       /usr/lpp/booksrv/cgi-bin/*
```

A directory may be protected. For example:

```
Protect /nucgi/*                  Prot_Resource
```

If you try to run a CGI program from this directory, you will get prompted by your browser to sign in (using your usual mainframe userid and password).

OUR DEVELOPMENT PROCESS

We'll follow a staged process:

1 Write a DB2/REXX program that works in TSO.

2 Get it into USS and run it from there.

3 Adapt it to run as a CGI program.

GETTING STARTED – DB2 REXX

We need a simple query that will produce some output with minimal code. This is the query:

```
SELECT DBNAME,COUNT(*) AS CT FROM SYSIBM.SYSTABLESPACE
     GROUP BY DBNAME
     ORDER BY DBNAME
     WITH UR;
```

It gives a list of databases with at least one tablespace, and a count of the tablespaces in each one. As usual with catalog queries, I've coded WITH UR so that I don't take out any page or row locks.

It’s easy to embed this within a REXX.

The first few lines will be much the same for all REXXes that use DB2:

```
/******************REXX*********************************/
/*    REXX to test REXX/DB2                             */
/*    Alan Smith September 2ØØ3                         */
/*******************************************************/
"SUBCOM DSNREXX"
if RC then
  myRC = RXSUBCOM("ADD","DSNREXX","DSNREXX")
```

After the comments, the **SUBCOM** command checks whether the DSNREXX environment is set up. If not, **RXSUBCOM** is called to do so.

Next:

```
ADDRESS DSNREXX
"CONNECT DB2S"
call check_sql_code("connect")
```

I set up DSNREXX as the default calling environment, connect to the required subsystem, and call a routine to check the SQL code returned. The routine is a pretty rudimentary one:

```
/*****************************************************/
/*    Routine to check sql code and exit if negative */
/*****************************************************/
check_sql_code:
if SQLCODE < Ø then
  do
  say "sql error " SQLCODE SQLERRMC
  exit 12
  end
return
```

The main part of the program sets up the SQL statement and cursor, opens the cursor, and loops round fetching rows. Each time through the loop, it prints out the name of the database and the count of tablespaces in it:

```
STMT = "SELECT DBNAME,COUNT(*) AS CT FROM SYSIBM.SYSTABLESPACE",
            " GROUP BY DBNAME",
            " ORDER BY DBNAME",
            " WITH UR;"
"EXECSQL DECLARE C1 CURSOR FOR S1"
```

```
"EXECSQL PREPARE S1 FROM :STMT"
call check_sql_code("prepare")
"EXECSQL OPEN C1"
call check_sql_code("open cursor")
"EXECSQL FETCH C1 INTO :DBNAME,:DBCOUNT"
do while SQLCODE ¬= 100
  call check_sql_code("fetch")
  say DBNAME DBCOUNT
  "EXECSQL FETCH C1 INTO :DBNAME,:DBCOUNT"
end
"EXECSQL CLOSE C1"
call check_sql_code("close cursor")
```

Finally, clean-up involves disconnecting from the subsystem and removing the host command environment:

```
"DISCONNECT"
call check_sql_code("disconnect")
myRC = RXSUBCOM("DELETE","DSNREXX","DSNREXX")
exit
Running the program outputs something like this:

D##MAP   2
DB2S     14
DCHORLEM 3
DCZDATA  1
DCZTEST  1
DDBIVP   7
DDBIVPSQ 6
.
.
.
```

It's important to get all this running without error within TSO, so any problems we come up with later will not be DB2/REXX ones.

Now it's time to get it working within USS.

MOVING INTO UNIX SYSTEM SERVICES

First we need to copy the REXX we created earlier into USS. We can do this easily with ISHELL.

Go into ISHELL. First create a new directory called **tests** to put the REXX in.

```
File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
--------------------------------------------------------------------------
```

```
                              OpenMVS ISPF Shell

Enter a pathname and do one of these:

    - Press Enter.
    - Select an action bar choice.
    - Specify an action code or command on the command line.

Return to this panel to work with a different pathname.
                                                                    More:     +
   /u/smithac/tests/
   ___________________________________________________________________________
   ___________________________________________________________________________
   ___________________________________________________________________________
```

Put in the full pathname as shown.

Then from the directory pull-down menu, choose **New**:

```
 Directory  Special_file  Tools
--------------------------------
 2  1. List directory(L)...
    2. New(N)...
    3. Attributes(A)...
    4. Delete(D)...
    5. Rename(R)...
    6. Copy to PDS(C)...
    7. Copy from PDS(I)...
    8. Print(P)
    9. Compare(M)...
   10. Find strings(F)...
   11. Set working directory(W)
   12. File system(U)...
```

When prompted for permissions, just press *Enter*. You've just created the tests directory. Now we'll copy in the REXX from our PDS.

Again from the directory pull-down menu, choose Option 7, **Copy from PDS**. You get a pop-up box asking you for the name of the PDS:

```
                    Copy a PDS into a Directory

Copying into directory:
/u/smithac/tests/

Enter PDS name:
'nutso.smithac.isprlib'_____________________
```

```
Select additional options:
_  Binary copy
/  Member list...
/  Make names lowercase
_  Conversion...

Permissions  . . . . . . . . . 775  (3 digits, each Ø-7)
Append suffix to names . . . . _____________________
```

Enter the name of the PDS to copy from, leave **Member list…** selected, and leave the permissions as shown. Press *Enter*. This gives you a member selection list:

```
               Member List

Select members to copy
   Member     Filename     Row 65 of 1Ø2
_  SORTNS     sortns
_  SPCSTF     spcstf
_  STRIPIT    stripit
_  SYSADM     sysadm
_  T          t
_  TEST       test
_  TESTARG    testarg
_  TESTB      testb
_  TESTD      testd
s  TESTDB2    testDB2.rx
_  TESTDSN    testdsn
_  TESTF      testf
_  TESTIT     testit
_  TESTIT2    testit2
_  TESTLDS    testlds
_  TESTLOG    testlog
_  TESTR      testr
Command ===> ____________________________________
```

I've chosen member TESTDB2, and I've added a *.rx* suffix to the target name. The suffix isn't necessary, but it gives me a clue as to what sort of code is in the file.

Press PF3 and *Enter* to get out of ISHELL.

Now we'll do everything else in OMVS. Type **TSO OMVS**. This takes you into the Unix shell and sets your current working directory to your home directory:

```
 - - - - - - - - - - - - - - - - - - - -
 -                                     -
 -  z/OS 1.2 Unix System Services      -
 -                                     -
 - - - - - - - - - - - - - - - - - - - -
/u/smithac >
```

Type **cd tests** to take you to the directory you created earlier:

```
/u/smithac > cd tests
/u/smithac/tests >
```

Then list what's in the directory:

```
/u/smithac/tests > ls -al
total 40
drwxrwxr-x   2 SMITHAC  NUDBS       8192 Sep 10 12:42 .
drwxr-x--    4 SMITHAC  NUDBS       8192 Sep 10 13:02 ..
-rwxrwxr-x   1 SMITHAC  NUDBS       3807 Sep 10 12:42 testDB2.rx
```

You can see the REXX, *testDB2.rx*, that we created earlier. Note the permission bits for the file (the left-most column). After the directory bit, they show that SMITHAC (me) has read, write, and execute access to the file (rwx); group NUDBS has the same access; everyone else has read and execute access (r-x). These are the permissions you want if you want people to be able to run the program. If the permission bits are different, you can set them to the value shown by specifying **chmod 775 testDB2.rx**.

You can run the REXX just by typing **testDB2.rx**. When I did this I found a problem with whatever process within USS was running the REXX – it started looping. Looking over at the SDSF DA queue, I found that a task under my name was using 25% of the mainframe's CPU! I had to get it cancelled. Using trial and error, I found that spacing out the asterisks on either side of **REXX** on the first line solved the problem (see full listing below). You can edit the REXX by typing **oedit testDB2.rx**.

If you get a NOT FOUND message when you try to run the program, either you've misspelt the program name, or the current directory is not in your path (PATH is an environment variable that tells the shell which directories to search for programs – it will vary between sites and users). To make it find your program, prefix the program name with **./** (a full-stop and a

slash). In other words type **./testDB2.rx** to execute the program.

You should then get another error message:

```
/u/smithac/tests > testDB2.rx
SUBCOM: FSUM7351 not found
     8 *-* "SUBCOM DSNREXX"
       +++ RC(127) +++
     9 +++  if RC
IRX0034I Error running ./testDB2.rx, line 9: Logical value not 0 or 1
```

SUBCOM seems to be a TSO command. Because the program isn't running in TSO, we get an error. Just deleting this and the next line in the REXX solves the problem.

You may or may not get this next problem:

```
/u/smithac/tests > testDB2.rx
     8 +++ myRC = RXSUBCOM("ADD","DSNREXX","DSNREXX")
IRX0043I Error running ./testDB2.rx, line 8: Routine not found
```

This time it can't find the DSNREXX module, which is in SDSNLOAD. I didn't have this problem when running the REXX in TSO because I've got SDSNLOAD allocated to my TSO session. If you've got SDSNLOAD in the linklist you should be OK. If not, there are a few ways around it.

First, you could add SDSNLOAD to the linklist – but you probably wouldn't want to make a global change just for this.

The second way is by typing the command **export STEPLIB=SYS2.DB2.SDSNLOAD**, which will work just for you and just for this session.

Thirdly, embed the export command in a shell script that calls the REXX. At the prompt, type **oedit testDB2.sh** to create a new file. Enter this text into the file:

```
export STEPLIB=SYS2.DB2.SDSNLOAD
testDB2.rx
```

This makes the STEPLIB environment variable available and then calls the REXX. The name of the SDSNLOAD dataset will probably differ at your installation.

Save this file and then run it. You should now have the same output that you got when running the REXX in TSO.

The final REXX should look like this:

```
/*                 REXX                                    */
/*    REXX to test REXX/DB2                                */
/*    Alan Smith September 2ØØ3                            */
/*********************************************************/
myRC = RXSUBCOM("ADD","DSNREXX","DSNREXX")
ADDRESS DSNREXX
"CONNECT DB2S"
call check_sql_code("connect")
STMT = "SELECT DBNAME,COUNT(*) AS CT FROM SYSIBM.SYSTABLESPACE",
            " GROUP BY DBNAME",
            " ORDER BY DBNAME",
            " WITH UR;"
"EXECSQL DECLARE C1 CURSOR FOR S1"
"EXECSQL PREPARE S1 FROM :STMT"
call check_sql_code("prepare")
"EXECSQL OPEN C1"
call check_sql_code("open cursor")
"EXECSQL FETCH C1 INTO :DBNAME,:DBCOUNT"
do while SQLCODE ¬= 1ØØ
  call check_sql_code("fetch")
  say DBNAME DBCOUNT
  "EXECSQL FETCH C1 INTO :DBNAME,:DBCOUNT"
end
"EXECSQL CLOSE C1"
call check_sql_code("close cursor")
"DISCONNECT"
call check_sql_code("disconnect")
myRC = RXSUBCOM("DELETE","DSNREXX","DSNREXX")
exit
/*********************************************************/
/*                                                       */
/*    Routine to check sql code and exit if negative     */
/*                                                       */
/*                                                       */
/*********************************************************/
check_sql_code:
if SQLCODE < Ø then
  do
  say "sql error " SQLCODE SQLERRMC
  exit 12
  end
return
```

So now we've got a REXX running in USS and contacting DB2. Now we'll move onto CGI.

A CGI PROGRAM

As I mentioned before, a CGI program is a program that writes out HTML. So let's start with a simple REXX program that doesn't do very much, just to test everything's in place and working. Then we'll merge this with our DB2 REXX.

Go to the directory where you'll put your CGI programs, and create a program called *test.rx*.

The HTML I'm going to output is this:

```
<HTML>
<HEAD>
<TITLE>Just testing</TITLE>
</HEAD>
<H1>Testing REXX</h1>
</BODY>
</HTML>
```

This just puts out a page with a line saying **Testing REXX** in a heading font.

The first thing the REXX must do is output the http header, followed by a blank line:

```
say 'Content-type: text/html'
say
```

It'll then output the rest of the HTML. The whole thing looks like this:

```
/* REXX                                                 */
/*    REXX to test REXX as CGI                          */
/*******************************************************/
say 'Content-type: text/html'
say
say
say '<HTML>'
say '<HEAD>'
say '<TITLE>Just testing</TITLE>'
say '</HEAD>'
say '<H1>Testing REXX</h1>'
say '</BODY>'
say '</HTML>'
exit
```

To test this, go into the Web browser on your PC. If the site is

known to the domain name server as nuibm, the CGI root directory is called nucgi, and we have created a directory called DB2 to put our CGI programs in, the URL you need to input is http://nuibm3/nucgi/DB2/test.rx.

ADDING DB2

We can now merge the DB2 REXX with CGI REXX. The easiest way is to take the DB2 REXX and add a routine to output the HTML at the start:

```
write_html_prologue:
say 'Content-type: text/html'
say
say
say '<HTML>'
say '<HEAD>'
say '<TITLE>Just testing</TITLE>'
say '</HEAD>'
say '<H1>Testing REXX</h1>'
say '<p><TABLE BORDER>'
return
```

Add another to output the HTML at the end:

```
write_html_epilogue:
say '</TABLE>'
say '</BODY>'
say '</HTML>'
return
```

Note that these routines are also outputting the tags to start and end a table.

The line which outputs the result of each fetch is changed slightly to output a row of a table:

```
say "<TR><TD>" DBNAME "<TD>" DBCOUNT
```

The whole thing now looks like this:

```
/*                 REXX                                  */
/*    REXX to test REXX/DB2                              */
/*    Alan Smith September 2003                          */
/*******************************************************/
call write_html_prologue
myRC = RXSUBCOM("ADD","DSNREXX","DSNREXX")
ADDRESS DSNREXX
```

Figure 1: Error

```
"CONNECT DB2S"
call check_sql_code("connect")
STMT = "SELECT DBNAME,COUNT(*) AS CT FROM SYSIBM.SYSTABLESPACE",
           " GROUP BY DBNAME",
           " ORDER BY DBNAME",
           " WITH UR;"
"EXECSQL DECLARE C1 CURSOR FOR S1"
"EXECSQL PREPARE S1 FROM :STMT"
call check_sql_code("prepare")
"EXECSQL OPEN C1"
call check_sql_code("open cursor")
"EXECSQL FETCH C1 INTO :DBNAME,:DBCOUNT"
do while SQLCODE ¬= 100
  call check_sql_code("fetch")
  say "<TR><TD>" DBNAME "<TD>" DBCOUNT
  "EXECSQL FETCH C1 INTO :DBNAME,:DBCOUNT"
end
"EXECSQL CLOSE C1"
call check_sql_code("close cursor")
"DISCONNECT"
call check_sql_code("disconnect")
myRC = RXSUBCOM("DELETE","DSNREXX","DSNREXX")
exit
```

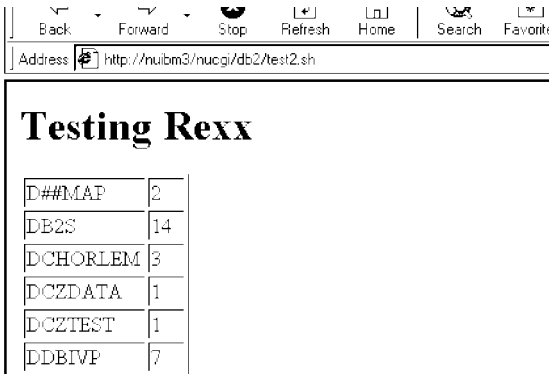

Figure 2: New result

```
check_sql_code:
if SQLCODE < Ø then
  do
  say '</TABLE>'
  say "<p>sql error " SQLCODE SQLERRMC
  exit 12
  end
return
write_html_prologue:
say 'Content-type: text/html'
say
say
say '<HTML>'
say '<HEAD>'
say '<TITLE>Just testing</TITLE>'
say '</HEAD>'
say '<H1>Testing REXX</h1>'
say '<p><TABLE BORDER>'
return
write_html_epilogue:
say '</TABLE>'
say '</BODY>'
say '</HTML>'
return
```

When running this, I found the same problem that I had before – see Figure 1.

Again, if you've got DB2 in the linklist, you won't have this problem. I can use the same technique as before – create a shell script which does the export and then calls the REXX:

```
export STEPLIB=SYS2.DB2.SDSNLOAD
test2.rx
```

The URL then becomes http://ibm3/nucgi/DB2/test2.sh.

There is another method – httpd.envvars is a file that sets the environment variables for the http server. You can embed the 'STEPLIB=' line in here – make sure that there isn't one already in there. The server needs to be restarted to pick up the change.

I used the shell script method, and eventually got Figure 2.

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

Alan Smith
Norwich Union (UK)

DB2 V7.1 CBPDO install

The following is the contents of a letter to IBM, sent in hopes of improving the CBPDO packaging provided to simplify software product installation.

In the hope of improving the process to install large z/OS-based software products like DB2, especially using the CBPDO process, I wanted to recount the many problems I ran into during my recent DB2 Version 7 installation. Some of these problems were reported to IBM at the time they occurred, but they are included here to give a more complete picture. It should also be noted that this was my first experience with the CBPDO process. And my first use of SMP/E in a decade.

DASD REQUIREMENTS – THE MEMO

Trying to determine the DASD requirements for DB2 Version 7 before the installation tape and paper Program Directory arrived was complicated by two things:

- The Program Directory on the IBM DB2 Web site was not the current revision (two revisions out of date).
- Most Program Directories quoted DASD space requirements in blocks, rather than tracks, but nowhere defined the assumed size of a block when calculating the values listed.

Once the five CBPDO installation tapes for DB2 7.1 arrived, the first big challenge was discovering that a recent CBPDO change meant that the Memo to Users is just a printed form telling you how to get the Memo to Users Extension off the Installation Tape.

Next, I found the change in tape file numbers with CBPDO very confusing. To access the Sample Jobs, I used the JCL in Section 6.1.4 (page 39) of the Program Directory. It shows LABEL=(6,SL), which I eventually figured out should be changed to 14, based on the Memo to Users Extension. It took a while longer to determine that File 14 was on the fourth tape.

SMS, NUCID

The first sample job used was DSNTIJAA. It especially, and many other jobs, could use a short section of comments indicating exactly what to delete for SMS-managed volumes. Admittedly, DFSMS will ignore VOL= and UNIT=, but it can be confusing.

Much more important was the NUCID(9) in the ADD OPTIONS statement. This is no longer supported in newer versions of SMP/E, and there should be comments both in the instructions in the job and beside the NUCID itself. Unlike most other z/OS software, SMP/E does not tolerate the obsolete NUCID, generating the following errors:

```
GIM57802E ** THE NUCID OPERAND IS NO LONGER ALLOWED FOR THE UCLIN
COMMAND.
GIM25601I    THE SPECIFIED ENTRY WAS NOT UPDATED BECAUSE OF AN ERROR
DURING UCLIN PROCESSING.
```

SPACE PROBLEMS

Still in DSNTIJAA, SMPPTS is coded with:

```
SPACE=(80,(48500,9250,300),,,ROUND)
```

which translates to 4.63 cylinders for the primary and less than one cylinder for the secondary, or 17.88 cylinders when it abends with 16 extents. Even without DB2 Utilities and DPROP, DB2 Version 7.1 with all maintenance included on the CBPDO tape requires an SMPPTS with 713 cylinders on a 3390 or equivalent. Depending on how you count it, that is either 150 times larger (one extent) or 40 times larger (16 extents) than the JCL was written for. I did a lot of tape remounts to figure that out.

I realize that this resulted from the huge amount of maintenance for DB2 Version 7.1. It is reasonable to assume that the CBPDO process should take that into account and update the JCL accordingly. I don't feel that listing the URL to http://www.s390.ibm.com/smpe/smppts.html in the Memo is enough.

In that same job, SYSUT1 (and likely all SYSUT*n*) was also too small:

```
ADD DDDEF(SYSUT1  ) NEW DELETE CYL SPACE(5,1) UNIT(SYSALLDA) .
```

had to be changed to:

```
ADD DDDEF(SYSUT1  ) NEW DELETE CYL SPACE(25,15) UNIT(SYSALLDA) .
```

Later, during the APPLY/ACCEPT process for the 'Additional FMIDs', SMPWRK3 ran out of space, E37-04 IEC032I. SMPWRK*n* for both distribution and target libraries were increased from 10 cylinders for the primary extent and 5 cylinders for the secondaries, to 100 and 30 respectively.

SDXXLOAD failed on too few directory blocks. It was set to 16. It and ADXXLOAD were set to 50. REDO RETRY(YES) COMPRESS(ALL) was added to the APPLY on the re-run.

TIME=, THE INSTALL PDS

Admittedly, the test system I was using had some WorkLoad Management problems at the time, but the size of DB2 makes for long RECEIVE and ACCEPT jobs, which requires the addition of a TIME= parameter for some job classes. I would recommend the addition of, or at least a comment to suggest the addition of, a TIME= parameter to the JCL EXEC statement. Based on my reading of the JCL manual, TIME= on the EXEC statement is the only way to increase the installation default for the job class; TIME= on the JOB statement can be used only to decrease the time limit.

A bit confusing, based on the number of datasets that exist (I ended up with 150 datasets), the sample JCL member DSNISMKD refers in both comments and JCL to hlqual.INSTALL as the standard name in the Program Directory for the sample JCL PDS. But the JCL in Section 6.1.4 on page 39 refers to it as jcl-library-name.

USS

As someone who had never used Unix System Services (USS) until this DB2 installation, I would have much preferred documentation in the Program Directory, and optional code in the

DxxISMKD, to give DB2 Version 7 its own HFS (in a separate z/OS dataset). On the same subject, the default USS paths for JDBC/SQLJ, IAV Extenders, and Text Extenders should be structured so that it is easy to have a single HFS for all three, but separate from the HFS for other DB2 Versions (ie Version 8 and 6). I would recommend:

- JDBC/SQLJ – /usr/lpp/db2/db2710/java/
- IAV Extenders – /usr/lpp/db2/db2710/iav/
- Text Extenders – /usr/lpp/db2/db2710/text/.

On the same subject, I found the -PathPrefix- parameter in DxxISMKD extremely confusing. For example, the statement in the comments 'FOR USERS INSTALLING IN THE ROOT, THIS WOULD BE '/'.' is misleading, since usr/lpp/db2… is appended to whatever value is specified for –PathPrefix-. I would argue that –PathPrefix- would be much more useful if it defaulted to /usr/lpp/db2/db2710/. And, obviously, if the REXX EXECs did not add the usr/lpp/db2….

MORE SMS

Sample JCL member DSNALLOC is the best example of a more widespread problem alluded to earlier: forcing the vast majority who are using SMS-managed volumes to worry about UNIT= and VOL=SER= values for DASD. Admittedly, the default UNIT value of SYSALLDA works without modification. Figuring out what values to code for VOL=SER= in each job that requires it is a needless distraction that took me a fair amount of time, when added up for all the jobs. It was further compounded by my inclination, when seeing the first job (DSNTIJAA), to try to remove all VOL= and UNIT= references for DASD. That quickly proved a thankless, frustrating, and futile exercise.

I would recommend providing two JCL libraries, one for SMS and one for non-SMS environments. Alternatively, commenting out all unit and vol references in such a way that a non-SMS environment could remove just those comment marks with one

or two ISPF Editor CHANGE commands that are well documented in the JCL comments.

ERROR PTFS

The biggest, in terms of time spent, problem was the CBPDO tape's inclusion of PTFs known to have errors. Only now, three months later, did I notice the CBPDO RIMLIB member SMPRERR, which appears to try to help with the issue. Call me a dummy, if you like, but it took me a man-week to figure out, by trial and error, and asking a lot of technical people a lot of questions, what SMP/E APPLY command would actually work properly when error PTFs are present. The supplied APPLY looks like this, and generated a huge number of GIM35949I messages:

```
APPLY  SELECT(
              HDB771Ø,
              HIY771Ø,
              HIZ771Ø,
              HDB771A,
              JDB7714,
              HIR21Ø1
              )
GROUPEXTEND
FORFMID(HDB771Ø,
        HIY771Ø,
        HIZ771Ø,
        HDB771A,
        JDB7714,
        HIR21Ø1
        )
BYPASS(HOLDSYS,HOLDUSER,
HOLDCLASS(UCLREL,ERREL)).
```

The final APPLY's BYPASS clause that actually worked looked like this:

```
BYPASS(HOLDSYS,HOLDUSER,
       HOLDERROR
HOLDCLASS(UCLREL,ERREL)).
```

Yes, it looks so simple, just add HOLDERROR, but I tried a large number of other possibilities before eventually getting this to work. I tried BYPASS(HOLDERROR) almost immediately, but without the other supplied BYPASS and HOLDCLASS specifications.

MISSING DDDEFS, PURGE, MORE MEMO

Along the way, there always seemed to be missing DDDEF entries – SCSQLOAD and SCLBCPP were the major examples.

Finally, just when I thought I was finished, having completed the ACCEPTs, we discovered that PURGE was the default OPTION in the SMP/E global zone, which meant I had to re-RECEIVE all of the maintenance so that the DBAs could determine what had been applied and what had not.

Overall, it is extremely confusing trying to work with both the Memo to Users Extension and the Program Directory. The Memo is essentially a huge list of revisions to the Program Directory. The ideal CBPDO improvement, to my thinking, would be to replace the Memo to Users Extension with generated Program Directory/ies for the product(s) shipped.

DPROP

At the same time, Version 7.1 of DB2 Utilities Suite and DataPropagator (DPROP) were also installed. A month later, a DBA asked me to investigate DPROP PTF UQ62505. Fortunately, I still had the SMP/E output from the DPROP install, or I would not have believed it. The PTF had been applied but SMP/E had then overlaid the original distribution onto the library members that the PTF had replaced. After much discussion in IBM ETR 62264,017,649, it was determined that the SMP/E coding was incorrect for this PTF, but this only creates a problem when the product is installed and the maintenance applied in the same SMP/E run. The PTF was coded as ++SAMP while the original distribution was ++MSG, and SMP/E failed to recognize that both were referencing the same member of the same library.

I was able to get the DPROP Labs to check that no other PTF was miscoded, but was unable to get them to fix the PTF or to include it in the DPROP V7 PSP bucket. I felt that both should have been done.

Jon E Pearkins
Adiant (Canada)

The UDB DB2BATCH facility

What is the DB2BATCH tool? Well, in a nutshell, it's a tool that allows you to run some SQL and gather related performance data that you can use in making comparisons. The manual calls this a benchmarking tool.

As always, the best way to look at how the tool works is to look at an example. Let's use the SAMPLE database, and run a query against the STAFF table. The following statements were run on a Windows 2000 system using the db2admin userid (install userid) running DB2 UDB 8.1 FP1.

First, create a file (sql01.txt) containing the following SQL:

```
SELECT * FROM Q.STAFF;
```

Now create a run file called db2b.bat that contains:

```
db2batch -d sample -f sql01.txt -i complete -o r 10 p 5 -r db2b.out1
```

You don't have to connect to the database before executing the bat file, the tool will do it automatically. Run this bat file as:

```
>db2b.bat
```

The first time you execute the db2batch command you will see a message as below, because you are binding the DB2BATCH.BND file:

```
Bind is successful.  Used bindfile: C:\PROGRA~1\IBM\SQLLIB\BND\
DB2BATCH.BND
Running in Embedded Dynamic mode.
```

Let's look at some of the parameters specified in the command. The **-d** parameter is the database where the table resides, and DB2BATCH will connect to this database. The **-f** parameter points to the file containing the SQL to execute. The **-i** parameter indicates how the SQL preparation, fetch, and execution times are displayed – I have specified COMPLETE, which means it shows these figures separately. The **-o** parameter means use

the control options that follow the **-o**. I have specified two such control options, namely **r** and **p**. **r** means return only this many rows to the result set, ie all the rows are still retrieved, but only the first **r** are written to the output file. The **p** control option is the performance detail option, which I set to 5. Lastly, the **-r** parameter means write the output to the file name specified following the parameter. If you wanted to write the summary of results to a separate file, you would specify this second output file on the **-r** parameter (as –r db2b.out1,db2b.out2). Then, the db2b.out1 file will contain the output from the query and the snapshot output (because the performance detail option was set to 5). The db2b.out2 file will contain a summary of the results. If you wanted to populate the EXPLAIN tables, then you would use the **e** control option and set it to either 1 or 2 (a value of 0 means don't populate the EXPLAIN tables). A value of 1 means don't run the query but populate the EXPLAIN tables, and, not surprisingly, a value of 2 means run the query and populate the EXPLAIN tables. All of these parameters are explained in the *Command Reference* (SC09-4828-00).

To make the output file(s) easier to read, use WORDPAD to read them.

If you wanted to, you could put the control options in the SQL file as shown below:

```
--#SET PERF_DETAIL 5 ROWS_OUT 10
SELECT * FROM Q.STAFF;
```

Then put the following in the db2b.bat file:

```
db2batch -d sample -f sql01.txt -r db2b.out1
```

The DB2BATCH command is a useful tool for monitoring the performance of an SQL statement over the course of time. You can run the command at regular intervals and compare the output files to see, for instance, whether or not the elapsed time has increased. Put the command in your armoury for monitoring your system to keep it in top condition.

C Leonard
Freelance Consultant (UK)

IBM has announced four new technologies that enable developers to deploy and manage applications based on DB2 UDB.

The Data Mining functionality of DB2 is now available for the first time in a development environment for software developers. WebSphere Studio Plug-in for DB2 Intelligent Miner can be used as a graphical front end for DB2 Intelligent Miner Scoring/ Modeling, which allows users to add data-mining functionality to their applications. The technology allows application developers to use the data-mining functionality of Intelligent Miner in their projects without having to learn Intelligent Miner's complex SQL API, hence making data mining accessible to software developers.

With WebSphere Studio Plug-in for DB2 Intelligent Miner, there is no need to switch from one mining tool to an application development tool. The mining process, which has to be integrated in an application, is designed in the development environment which is used to build the whole application. Furthermore, since programmers now have visual support to design their mining process, they don't have to learn the API of the mining tool as they did in the past.

An effective monitoring solution is essential to support error-free operation. The new Replication Monitoring Center for DB2 (RMCDB2) supports error prevention, minimization of down time, and problem resolution.

IBM has released a free trial version of URI Access for DB2, a solution that enables faster and easier access to data stored in DB2 from many environments such as Web browsers, XML parsers and other web-based clients.

The Easy Mining Procedures for DB2 Intelligent Miner facilitates has an SQL API for data mining.

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

* * *

Cognos has launched Cognos ReportNet, which offers a Web-based enterprise query and reporting environment for DB2 and AIX users.

The combination of Cognos ReportNet with DB2 UDB provides data access tools integrated with an automated data management process. ReportNet, with DB2 UDB, can help organizations uncover hidden relationships, spot key trends, and plan more effective strategies using all their business data.

For further information contact:
Cognos, 3755 Riverside Drive, PO Box 9707, Station T, Ottawa, ON, Canada K1G 4K9.
Tel: (613) 738 1440.
URL: http://www.cognos.com/products/businessintelligence/reporting/reportnet.html

* * *